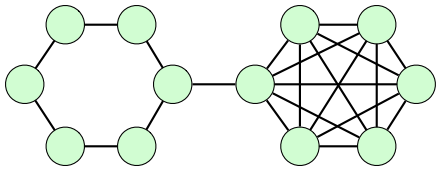
1 Large-scale structure in networks

Most of the network measures we have considered so far can tell us a great deal about the shape of a network. However, these measures generally describe patterns as if they were spread uniformly throughout the network. For instance, reciprocity tell us the average number of bidirectional links in a directed network; the clustering coefficient gives the average number of closed connected triples attached to a vertex; etc. As a result, these measures can hide heterogeneity in the distribution of these structures across the network.

For instance, recall the ring-clique network from earlier in the semester:



which as a clustering coefficient of $C = 60/73 = 0.82$. However, nearly all of this high value comes from the clique, which is the only part of the network with any triangles. By dividing the network into two parts, one containing the ring and one containing the clique, this heterogeneity would be more easily revealed.

Vertex level measures can provide some insight into structural heterogeneity, but they suffer from the opposite problem that the network-level measures do: they disaggregate the heterogeneity too much, and thereby can hide the tendency of vertices with similar measures to be found close to each other in the network.

What we seek, instead, is to identify and extract the large-scale organizational patterns of structural heterogeneity within networks. When networks are small, these patterns can often be identified by eye alone.[1] Even for networks of only a few hundred vertices, however, these patterns quickly become hidden behind the visual mess of edges on the screen, and we must instead rely on quantitative tools to pick them out. This is especially true for very large networks, composed of thousands or millions of vertices, where there is room for many types or scales of organization.

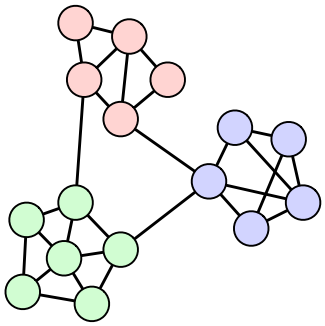
A useful analogy for this idea is that of clustering in vector data, in which points within a cluster are closer or more similar to each other than they are to points in other clusters. In this way, heterogeneity at the network-level could simply be the byproduct of mixing or averaging across

[1] Nearly all network visualization algorithms are force-directed graph drawing techniques, the most common of which is the Fruchterman-Reingold algorithm. For a good introduction to these techniques, see the MathWorld page `http://bit.ly/b1mo3y`.

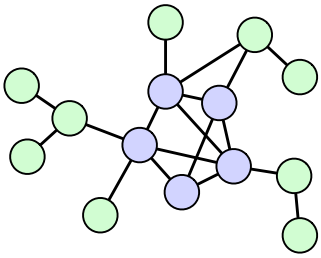
several distinct parts of the network that are themselves relatively homogeneous. This would be indicative of a pattern *modular organization* or *community structure*. In biological networks, such structures may be functional clusters of genes or proteins, and they might represent targets of natural selection above the level of individual genes but below the level of the whole organism. In social networks, they could be human social groups with common interests, histories or behaviors.

Community structure alone implies only a single level of organization, with relatively little structure within or between communities. But why stop at one level of organization? The natural generalization of community structure is called of *hierarchical* community structure, in which vertices divide into groups that further subdivide into groups of groups, and so forth over multiple scales. This structure is often described using a *dendrogram* or tree that shows how the smallest groups are nested within larger groups, and they in turn are nested within still larger groups, etc. A common assumption for this kind of large-scale organization is that two vertices who are "closely related", that is, are separated by a short distance on the tree, are more likely to be connected. This is typical, for instance, of human social structures, for instance: our class sits within the larger Computer Science department, which itself sits inside the larger College of Engineering, which sits inside CU, which sits inside Colorado schools, etc.

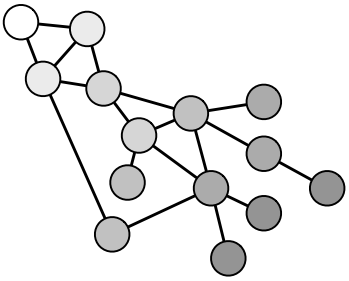
Another kind of large-scale structure is a *core-periphery* pattern, in which a dense core is surrounded by a more diffuse, weakly interconnected periphery. This notion has much in common with traditional measures of centrality, which can be viewed as proxy measurements for identifying "core" vertices. Similarly, an *ordered* pattern is one in which vertices arrange themselves roughly into a linear hierarchy, such that vertices at some position in the ordering tend to connect only to those lower in the ordering.



modular*



core-periphery



ordered

Caveats. There are many variations and subtleties in these terms, and they are not completely distinct patterns. For instance, vertices may have multiple, possibly even fractional, community memberships, a pattern called overlapping community structure. Similarly, core-periphery structure

can be thought of as a kind of modular structure, in which the core is an assortative community, meaning it is internally dense, the periphery is disassortative, meaning internally sparse, and the two are connected sparsely. Furthermore, an ordered structure is a kind of one-dimensional core-periphery structure, with multiple layers of periphery. Finally, in a sufficiently large network, these structures can be mixed or combined, sometimes in non-obvious ways, or they may appear at different scales, e.g., a network may be modular at the largest scale, but within-module structure might be ordered, or vice versa.

2 Community structure

Much of the past work on studying modular or community structure in networks has been in two directions. The first and much larger is in developing algorithms for identifying the presence of such structures from interactions (edges) alone. The second direction is in understanding the impact that the presence of modular structures in a network has on the dynamics of certain processes such as the diffusion of information or a disease across a network. In both areas, modules or communities are typically defined vaguely as largeish groups of vertices that are more densely connected to each other than they are to the rest of the network. Different researchers often formalize this loose notion in different ways, which can make results and algorithms difficult to compare. In this course, we will cover two popular approaches, modularity maximization and the stochastic block model.

2.1 Network partitions

Most community detection methods can be thought of as trying to solve the following problem. First, assume some *score function* f that takes as input a network and a *partition* of its vertices (defined below) and returns a scalar value. Now, find the partition P that maximizes f. The idea is that f encodes our beliefs about what makes a *good* partition with respect to representing modular structures. If f gives higher scores to partitions that place more edges within groups than between them, then maximizing f over all partitions yields the partition with the densest possible groups under f.

Network partitions are simply a division of a network into k non-empty groups (modules), such that every vertex v belongs to one and only one group (module). For a given value of k, the number of possible such partitions is given by the Stirling numbers of the second kind

$$S(n,k) = \left\{ \begin{array}{c} n \\ k \end{array} \right\} = \frac{1}{k!} \sum_{j=0}^{k} (-1)^j \binom{k}{j} (k-j)^n \ . \tag{1}$$

For instance, there are $S(4,2) = 7$ possible partitions of a network with $n = 4$ nodes (indexed as 1,2,3,4) partitioned into $k = 2$ groups:

$$\{1\}\{234\},\ \{2\}\{134\},\ \{3\}\{124\},\ \{4\}\{123\},\ \{12\}\{34\},\ \{13\}\{24\},\ \{14\}\{23\}\ .$$

The number of all possible partitions of a network with n vertices into k non-empty groups is given by the nth Bell number, defined as $B_n = \sum_{k=1}^{n} S(n,k)$, which grows super-exponentially with n.[2] That is, the universe of all possible partitions of even a moderately sized network is very very big and an exhaustive maximization of f is not typically feasible.

Instead, most algorithms use some kind of heuristic[3] for estimating the maximum of f.

2.2 Modularity maximization

Recall that the modularity function Q is a measure of assortative structure on enumerative vertex attributes. When we are given a set of labels on the vertices, we can calculate Q, which tells us the degree to which edges reflect a preference for appearing between like labels. If we treat these labels as hidden or "latent" variables, i.e., variables that we don't observe directly, then we can use the modularity function to search over possible labelings in order to find one that yields the highly assortative communities.

A brief review of modularity. Let k_i denote the degree of vertex i, c_i denote the index of the group containing vertex i, and define a function $\delta(c_i, c_j) = 1$ if vertices i and j are placed in the same group (i.e., if $c_i = c_j$) and 0 otherwise, then the modularity score of a partition is defined as

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \ . \tag{2}$$

This summation ranges over all pairs of vertices, but only accumulates value when the pair i, j are in the same community. That is, it only counts the internal edges of a community identified by the given partition. Crucially, the inner term counts these internal edges but subtracts their expected number under a random graph with the same degree sequence (this is the familiar configuration model; given that two vertices have degrees k_i and k_j, the probability that they are connected under the configuration model is exactly $k_i k_j / 2m$).

Equivalently, and perhaps more intuitively, the modularity score can be rewritten as a summation over the structure of the modules themselves

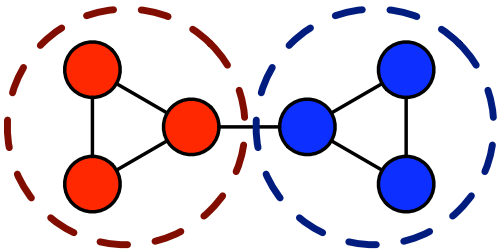
$$Q = \sum_{i} \left[\frac{e_i}{m} - \left(\frac{d_i}{2m} \right)^2 \right] \ , \tag{3}$$

[2] The first 20 Bell numbers (starting at $n = 0$) are 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, 678570, 4213597, 27644437, 190899322, 1382958545, 10480142147, 82864869804, 682076806159, 5832742205057.

[3] In Computer Science, the term "heuristic" is usually applied to any procedure that does not come with guarantees about its correctness, i.e., its ability to return the global maximum. This does not mean the heuristic is bad, only that we do not have a proof of its general correctness. The term "algorithm" is usually reserved for procedures with such guarantees. Outside of Computer Science, heuristics are nearly always called algorithms.

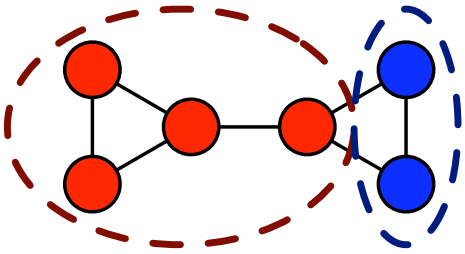
where now i indexes modules, e_i counts the number of edges within the ith module and d_i is the total degree of vertices in the ith module. Thus, the first term e_i/m measures the fraction of edges within module i and $(d_i/2m)^2$ is the expected fraction under the configuration model. A "good" partition—with Q closer to unity—represents groups with many more internal connections than expected at random; in contrast a "bad" partition—with Q closer to zero—represents groups with no more internal connections that we expect at random. Thus, Q measures the cumulative deviation of the internal densities of the identified modules relative to a simple random graph model.[4]

Recall our simple example for calculating the modularity: a simple network of two triangles connected by a single edge.



$Q_{\text{good}} = 0.357$

M_{good}	red	blue
red	6	1
blue	1	6



$Q_{\text{bad}} = 0.122$

M_{bad}	red	blue
red	8	2
blue	2	2

Where the M matrices give the counts of the number of edge stubs that run from vertices in community i to vertices in community j, and the sum over M is $2m$. Reading values from these matrices, the first partition has $e_i = 3$ and $d_i = 7$ (for both modules), which yields $Q = 5/14 = 0.357$; the second has $e_1 = 4$ and $d_1 = 10$ while $e_2 = 1$ and $d_2 = 4$, which yields $Q = 6/49 = 0.122$. Thus, dividing the network into two groups, each which contains one of the two triangles, identifies more assortative communities than the alternative.

Recall, however, that there are B_n possible partitions of a network with n vertices (in this case, $B_6 = 203$, which is not so large), which means that we must typically search non-exhaustively over the space of all partitions to find those that yield high modularity scores. In general, it's

[4]It should be pointed out that a high value of Q only indicates the presence of significant deviations from the null model of a random graph. A large deviation could mean two things: the network exhibits (i) genuine modular structure in an otherwise random graph, or (ii) other non-random structural patterns that are not predicted by a random graph. See Section 2.4 for some brief discussion of what this means for practical applications.

known that maximizing Q is NP-hard but many heuristics seem to work well in practice. Some of these include greedy agglomeration, mathematical programming (linear and quadratic), spectral methods, extremal optimization, simulated annealing and various kinds of sampling techniques like MCMC. We won't cover any of them in detail; if you're interested, I can point you toward several recent review articles on them.

Many of these algorithms run quite slowly on real-world networks, taking time $O(n^2)$ or slower; a few run very quickly, for instance, taking time $O(n \log^2 n)$ and can thus be applied to very large networks of millions or more vertices. Because they are heuristics, of course, faster algorithms must make more aggressive assumptions about how to search the partition space and thus are less likely to find the globally optimum partition.

2.3 A myriad of approaches

There are, of course, many other approaches to identifying communities in networks. Some are different ways of searching over partitions, but which use the modularity function to evaluate them. Otherwise use different score functions. We won't cover any of these in detail, except for the stochastic block model, which we will cover next. There are at least two qualitatively different approaches to clustering networks, which are worth mentioning.
The first is the *local* clustering algorithms. Note that the two approaches described above both require us to explicitly know the entire network structure. Thus, they require access to *global* information, e.g., the total number of edges in the network. Local techniques make no such assumptions and can be applied to networks like the WWW, which are too big to be fully known. Local methods often "crawl" outward from a particular starting point to identify the first "community" that encloses the starting vertex by looking for a boundary of vertices with relatively sparse connectivity to the rest of the network.

The second is the "soft" clustering algorithms for identifying *overlapping* communities, i.e., rather than sorting vertices such that they appear in one and only one cluster ("hard" clustering), vertices are allowed to belong to multiple communities, potentially with different strengths of association.

2.4 Three ill omens

Finally, there are many caveats about community detection. Here are three.

The first ill omen is the observation that despite a tremendous amount of activity on identifying clusters in networks, for the most part, we have no good explanations of what social, biological or technological processes should cause clusters to exist in the first place, or what functional role they play in those systems. (Presumably, these two things are related.) The best we have is a verbal argument due to Herbert Simon for why modular designs a good way to construct complex

systems. The argument goes like this:

Suppose we are tasked with assembling many small, delicate mechanical parts into a beautiful Swiss watch, a masterpiece of complexity. And, suppose that we are interrupted every now and then, perhaps by the arrival of email or the need to have a snack, and than if we are interrupted before we have fully completed the assembly of a single watch, the partially assembled watch falls to pieces and we must start afresh after our break is over. If, on average, we experience one or more interruptions during the long process of assembling a watch, we will never produce very many watches. Now, however, consider the advantage of a modular design to the watch. In this case, the final watch is produced by combining smaller groups of parts or modules. If these modules are themselves stable products, then now we can tolerate interruptions and still produce watches because at worst, we only lose the work necessary to build a piece of the whole, rather than the whole itself. Thus, it seems entirely reasonable to expect that, in the face of "interruptions" of all different kinds, complex systems of all kinds will exhibit a modular or even hierarchical organization.

A second ill omen comes from recent observations that many of the popular global techniques for identifying modules in networks exhibit two undesirable properties: (i) *resolution limits* and (ii) *extreme degeneracies*. The first observes that the technique cannot detect intuitively coherent communities that are smaller than some preferred scale. In general, the mathematics showing the appearance of this behavior always points to the same root cause, which is the random-graph assumptions built into the techniques' score functions. The second observes that there are an exponential number of alternative, high-scoring partitions, which makes both finding the best partition computationally difficult (in some cases, NP-hard) and interpreting the particular structure of the best partition ambiguous. These two properties make it somewhat problematic apply network clustering techniques in contexts where what we want is to find the *true* but unknown modules. For much more detail about these issues, see Good et al., "The performance of modularity maximization in practical contexts." *Physical Review E* **81**, 046106 (2010).

A third ill omen comes from the world of spatial clustering, a very old and still very active field. Many of the techniques for finding clusters in networks have antecedents in this domain. The general problem can be framed like so: we are handed a data set $\mathbf{x} \in \mathbb{R}^{\mathbf{n}}$ and our task is to identify the "natural" clusters of these data such that points in a cluster are closer (via some measure of distance) to each other than they are to all the other points.

However, in a 2002 paper titled "An Impossibility Theorem for Clustering", Jon Kleinberg showed, using an axiomatic approach, that no clustering function f can simultaneously satisfy three highly reasonably and practically banal criteria:

1. *Scale Invariance*: For any distance function d and any $\alpha > 0$, $f(d) = f(\alpha \cdot d)$. Intuitively, this requirement says that if we rescale all the distances between points by some constant factor

α, the best clustering should stay the same.

2. *Richness*: Range(f) is equal to the set of all partitions of $\mathbf{x}$, which simply requires that all possible partitions of our data be potentially okay candidates.

3. *Consistency*: Let d and d' be two distance functions. If $f(d) = \Gamma$, and d' is a Γ-transformation of d, then $f(d') = \Gamma$. In other words, if a particular distance function d yields the clustering Γ, then if we make a new distance function d' that reduces all of the distances arising within clusters while increasing all those distances between clusters—that is, we make the clusters internally more dense and move them further away from each other—then we should get the same clustering Γ under d'.

Thus, any practical solution to identifying clusters in spatial data can, at best, only have two of these properties. One consequence of this fact is that we can categorize clustering techniques into broad categories, based on which single criteria they violate. It's unknown if a similar impossibility theorem exists for network clustering.[5]

3 At home

1. Read Chapter 11.2–11.4 & 11.6–11.11 (pages 354–364 & 371–391) in *Networks*

2. Peruse Chapter 9 (pages 423–455) in *Pattern Recognition*

3. Next time: more block models

[5]It is not hard to see that the second and third criteria have natural analogs in the context of network clustering. The first criteria, however, has no clear analog.