# Networks and Community Detection

Joshua Mankelow

May 8, 2022

**Abstract**

Networks are an example of a rigorous model for analysing and understanding real world complex systems. A very important quality of these network models is the naturally emergent community structure. Community detection allows us to identify clusters in the network that are well connected amongst eachother. If a network is being used to model a real world system then finding this structure has many implications about the behaviour of the system such as predicting the structure of a network after a change in dynamics. In this essay I will discuss some simple methods for community detection in networks and their applications to the analysis and understanding of real world complex systems.

# Contents

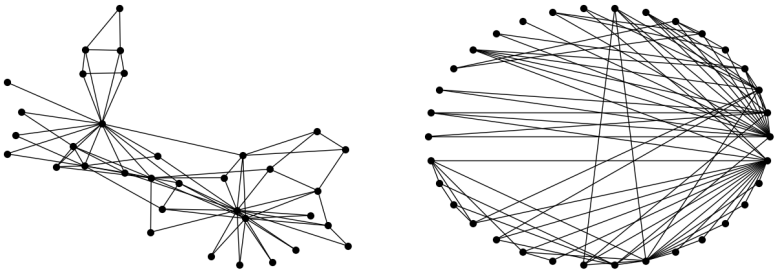
# 1 Introduction to Networks

Networks are considered as the combination of two separate objects - a set of nodes (vertices) and a set of links (edges) that connect nodes. The idea is to define a structure that can represent a set of things and how they're connected amongst each other. It turns out that this idea is invaluable for modeling real world systems. Examples of such real world systems include *Technological Networks*, *Social Networks*, *Information Networks* and *Biological Networks* [17, Contents]. A brief example of a network would be something like the following: imagine you and a number of people you speak to regularly are represented as dots (nodes or vertices) on a piece of paper. Then if any two people are friends, the dots representing those people are connected by a line (edge). If you then repeat this process by asking your acquaintances to list all their friends and so on, you will end up with a simple model of a *social network*.

With this model, it is easy to identify and detect any natural structure that emerges which we can then use to develop an understanding of the behaviour of the real world system that the network represents. The structure that I will explore in this essay is that of *communities*. Generally speaking, communities are subsets of a network that are *densely connected* amongst themselves, i.e. there is some notion of any node within a community being more closely connected to other nodes in the community than nodes outside the community on average. Communities are of particular interest due to the ubuquitous nature of networks and the predictive power that detecting such macroscopic structure has. However, before we dive into the details of communities and detecting them, I wish to provide some motivation, by way of example, of the kinds of situations that networks can arise in and why they are the natural model for the related systems.

## 1.1 Social Networks

To better illustrate the simple notion of a social network mentioned above, I will introduce the canonical community detection example of *Zachary's Karate Club*. Zachary's Karate Club is a dataset where "the data was collected from the members of a university karate club by Wayne Zachary in 1977. Each node represents a member of the club, and each edge represents a tie between two members of the club." [1, Metadata] In Figure 1, there are two different renderings of the Zachary Karate Club. Figure 1a shows the network rendered using a "spring" layout (which is a type of force directed graph drawing[13])and figure 1b shows the network rendered using a "circle" layout. These different layouts show us different parts of the underlying structure of the network. For example, in Figure 1a, it is clear which nodes in the network have the highest degree and which are of lower degree. It also facilitiates the observation of some of the community structure in the network. Meanwhile, in Figure 1b, it is much easier to see which nodes edges in the network would need to be removed to disconnect the network in a minimal way. The reason this dataset is the canonical example of community detection is that the question that comes with it is the following: Suppose two members of the club have a disagreement which causes the club to split in two. How does the club split? In Zachary's original paper on the topic, *An Information Flow Model for Conflict in Small Groups* [26], Zachary uses community detection techniques to predict how the network will split after



(a) Spring Layout. (b) Circle Layout.

Figure 1: Two renderings of the Zachary Karate Club network using data from KONECT.cc [14] and a Python library NetworkX [9].

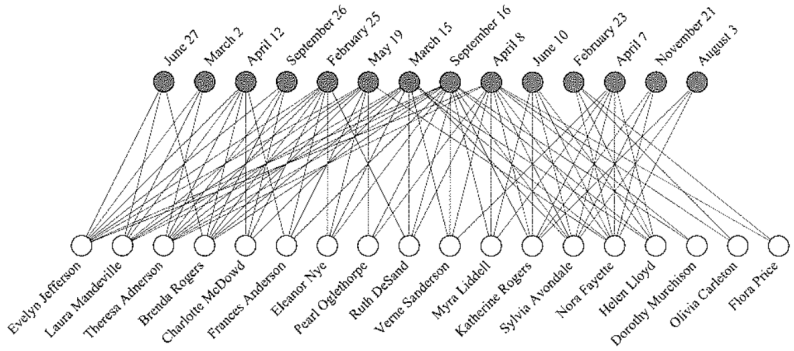



Figure 2: A rendering of the Southern Women Dataset from Newman [17, 39].

the disagreement. Out of 34 individuals, Zachary correctly predicts how 33 of them will choose a side after the disagreement.

There exist different ways to represent social networks and the representation you choose depends on the question you are trying to answer. For example, one might imagine having two types of nodes in a network. One type of node will represent a person and another type of node will represent an event. An edge is drawn between a person and an event if a person attended a given event and person A is considered connected to person B if they both attended the same event. One such example of this is the *Southern Women Dataset* [5]. This dataset is another example of a community detection problem because, after analysis of the data, it was found that women in the group were split into two discrete subgroups. This is significant because it highlights that even without knowledge of how individuals interact, we can still predict who they're likely to interact with.

## 1.2 Technological Networks

As a result of our intensely and digitally connected world, technological networks are of significant interest to researchers. The easiest example to consider is the Internet. The internet consists of many computers all connected by copper or fibre cables which signals are sent through to transmit data. As one might imagine, in the model, the computers are nodes and the cables are the edges. The internet needs to be robust against software and hardware failures and this is where the idea of commnity detection can help us. Saying that we want the internet to be robust is the same as saying that we want every node in the network to be strongly connected to every other node i.e. the number of possible routes between any two nodes is large. This means that, in the philosophy of community detection, we want the internet to act as one large community rather than multiple smaller communities that are loosley connected. An alternative way of looking at this is that once we've managed to identify the communities, we can then figure out which edges and nodes are the critical ones that allow passage from one community to another. This allows us to reinforce those edges and nodes to reduce the potential for failure.

Yet another example of a technological network would be the UK Power Grid. Network theory is a useful model here since for the UK Power Grid we're trying to solve exactly the same problem as with the internet — we want the system to be robust against hardware or software failures.

## 1.3 Information Networks

The most accessible example of an information network is that which is generated by looking back through the citations on a paper recursively. If Paper A references Paper B, then we will draw a directed edge connecting Paper A to Paper B. This will generate a network that shows which papers are referenced by which other papers and how information is reused. Applying community detection to such a network would show us the different academic working groups and perhaps even different fields or subfields of a subject. An example of such an analysis is given by Redner [22].

Another example of an information network is the World Wide Web which differs from the internet in that it refers to the webpages hosted on the internet rather than the servers and cables themselves. Mapping the world wide web as a network shows us communities of websites that regularly reference each other. The idea of modelling the world wide web as a network has been used by companies like Google to develop tools like PageRank to enable easier browsing of the web [20] and it also allows us to get an understanding of the topology of the web as a whole [2].

# 2 Properties of Networks

Community detection relies on us having sufficient information about the underlying structure of a network and to do that we have to understand its properties. This chapter will establish a more formal understanding of networks and will highlight some key properties and methods that we will use to exctract value about community structure later. This chapter also serves a dual purpose: to get the reader thinking about networks formally and all of the different ways that we can interact with them. As such, not everything mentioned in this chapter will be referred to later, but still provides value in the form of thought exercise.

## 2.1 Basic Definitions

To perform any mathematical operations on a network, we must first formally define what it is we mean by a network. The following definitions are inspired by Lambiotte [15] and Newman [17].

**Definition 1.** *(Undirected network) Let $V$ be a set of vertices (nodes) and let $E$ be a set of pairs of vertices such that if $e = (x, y) \in E$, then $x, y \in V$. An edge $e = (x, y) \in E$ is said to join $x$ and $y$ and $y$ to $x$. An **undirected network** is the pair $(V, E) =: N$. .*

The undirected network is the simplest type of network and on its own has interesting enough properties. However, for the sake of example and application, we will also introduce some other types of network that allow for more detailed models.

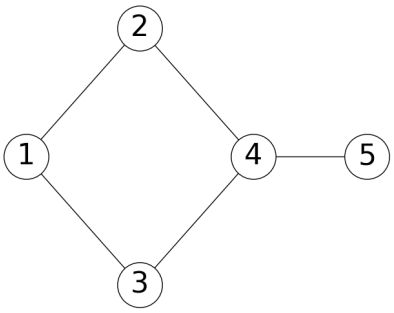
**Definition 2.** *(Directed network) Let $V$ be a set of vertices and let $E$ be a set of pairs of vertices such that if $e = (x, y) \in E$ then $x, y \in V$. An edge $e = (x, y) \in E$ is said to join $x$ to $y$, but if $x$ is joined to $y$ then $y$ is not necessarily joined to $x$. A **directed network** is the pair $(V, E) =: N$. .*

The intuition for directed graphs, is that edges may only be travelled along in one way. This comes in handy for modelling more intricate systems. The final network type of interest is that of the weighted network.

**Definition 3.** *(Weighted network) Let $V$ be a set of vertices (nodes) and let $E$ be a set of triples $V^2 \times \mathbb{R}$ such that if $e = (x, y, w) \in E$ then $x, y \in V$. The value $w$ is said to be the weight of the edge and the tuple $(V, E) =: N$ is called a **weighted network**.*

The objects defined above are useless without a rigorous way of mathematically representing them. To that end, we have to come up with a way of describing a network mathematically. This leads us to the definition of the adjacency matrix:

**Definition 4.** *(Adjacency matrix) Let $N = (V, E)$ be a network and label every vertex $v \in V$ with a number from 1 to $n := |V|$. The **adjacency matrix** of a network is the matrix of elements such that $(A)_{ij} = a_{ij} = 1$ if $(i, j) \in E$ and $a_{ij} = 0$ if $(i, j) \notin E$. In other words, if nodes $i$ and $j$ are connected by an edge in the network, then the corresponding element in the matrix is 1. Otherwise, it is 0.*




(a) Simple graph.

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

(b) Adjacency matrix.

Figure 3: A simple network and its adjacency matrix.

The adjacency matrix gives us our first way of representing a network. Figure 3 shows a basic example of a network and its associated adjacency matrix. This will form the basis for most of the analytical work we do going forwards. It's worth noting that there are also different types of adjacency matrix corresponding to the different types of network. For example, in the case of a directed network we will have a non-symmetric matrix where $a_{ij} = 1$ if $(i, j) \in E$ but $a_{ji}$ is not necessarily equal to 1. We also get something similar for weighted networks where we set $a_{ij} = w$ where w is the weight of the edge connecting $i$ and $j$ in $N$.

## 2.2 The Network Laplacian

The Network Laplacian is a simple extension of the adjacency matrix with more interesting properties.

**Definition 5.** *(Network Laplacian) The* ***Laplacian*** *of a network $N = (V, E)$, denoted by $L$ is given by the following:*

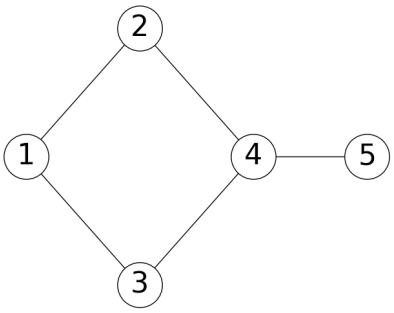
$$L = D - A,$$

*where $A$ is the adjacency matrix of the network and $D$ is a diagonal matrix containing the degrees of each vertex in the network such that $d_{ii} = deg(v_i)$ and $d_{ij} = 0$ if $i \neq j$.*

Figure 4 shows the same simple graph as before and its Laplacian matrix. Of course the largest difference between the laplacian and the adjacency matrix is the presence of the degrees of the nodes on the diagonal. This lets us encode all the data pertaining to a network in one matrix without any extra operations. Both the adjacency matrix and the laplacian are used regularly and each one has it's advantages and disadvantages. One such advantage of the adjacency matrix is that it's very easy to determine if there is a path between two nodes just via matrix multiplication.

## 2.3 Paths

When analysing a network, there is often great interest in which vertices can be reached from a given vertex. As such, we become interested in the idea of a




(a) Simple graph

$$\begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 2 & 0 & -1 & 0 \\ -1 & 0 & 2 & -1 & 0 \\ 0 & -1 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

(b) Laplacian

Figure 4: A simple network and its Laplacian.

path. A path in a network is defined in the following way:

**Definition 6.** *(Path) Let $N = (V, E)$ be a network. A **path** is a sequence of vertices $v_1, \ldots, v_n \in V$ such that $(v_i, v_{i+1}) \in E$ for all $i = 1, \ldots, n-1$. In other words, a path is a sequence of vertices such that every consecutive pair of vertices is connected by an edge in $E$. We say that the length of a path is the number of edges $(v_i, v_{i+1})$ that are traversed by the path. Note that under this definition, we may pass through each vertex in the network more than once.*

Paths are an important concept in community detection as they allow us to phrase questions in rigorous terms as opposed to loose concepts of connectedness. Paths also give us our first look into the usefulness of the adjacency matrix. Using the adjacency matrix, it is very simple to determine whether there exists a path between two vertices $i$ and $j$. Paraphrasing Newman [17, 137], suppose our adjacency matrix is given by A. If $i$ and $j$ are directly connected then $A_{ij} = 1$ and we are done. If $A_{ij} = 0$ then pick some $k$ such that $A_{ik} = 1$. Then it is simple to see that if $A_{kj} = 1$ then $A_{ik}A_{kj} = 1$ which implies that $i$ and $j$ are connected via $k$. In fact, we can even go so far as to calculate the total number of ways to draw a path of length two between $i$ and $j$, $N_{ij}^{(2)}$, in the following way:

$$N_{ij}^{(2)} = \sum_{k=1}^{n} A_{ik}A_{kj} = [A^2]_{ij},$$

where $[\cdot]_{ij}$ denotes the $(i, j)$-th element of the given matrix. Clearly, this process generalises to paths of arbitrary length $r$ and we can see that

$$N_{ij}^{(r)} = [A^r]_{ij}.$$

Also note that this solution counts each path but going in opposite directions. For example, you might have a path going $1 \to 4 \to 5 \to 2 \to 1$ which will also get counted separately by this method as the following $1 \to 2 \to 5 \to 4 \to 1$. This result isn't very useful, but it goes to show that the adjacency matrix we introduced before is useful and provides insight about the structure of our network.

Furthering our work on paths, we call a path that starts and ends at the same place a loop and we can calculate the number of loops of length $r$ using the spectral properties of the adjacency matrix. Paraphrasing Newman again [17, 137], our adjacency matrix $A$ can be written as $A = UKU^T$ because $A$ is symmetric meaning that it has $n$ real and non-negative eigenvalues with real valued eigenvectors. In this form, $U$ is our matrix of eigenvectors and $K$ is the diagonal matrix containing the eigenvalues. We know that $A^r = (UKU^T)^r = UK^rU^T$ and then the number of loops is given by

$$\begin{aligned} L_r &= \mathrm{Tr}(UK^rU^T) = \mathrm{Tr}(U^TUK^r) = \mathrm{Tr}(K^r), \\ &= \sum_i k_i^r, \end{aligned}$$

where $k_i$ is the $i$-th entry of the matrix $K$. There exist analogous results for all the different types of networks which Newman discusses further[17, 138]. Typically, we are interested in types of path known as *geodesic paths*.

**Definition 7.** *(Geodesic Path) A geodesic path (more commonly referred to as a shortest path) is a path through a network such that no shorter path exists.*

Geodesic paths are more interesting than general paths as they are necessarily self-avoiding as any time a path intersects with itself it adds unnecessary length. Geodesic paths are also used to define some other properties of networks such as the *diameter* and they typically give us a much better understanding of the structure of a network than regular paths do. Generally speaking, when the word "path" is written in this essay, it refers to a geodesic path.

## 2.4 Components

Components are a natural consequence of the notion of paths. Simply put, components are sets of vertices in the graph that are all connected to each other via paths.

**Definition 8.** *(Component) A component is a subset $C$ of the vertex set $V$ such that if $v_1, v_2 \in C$, then $v_1$ and $v_2$ are connected by a path. Furthermore if $v_3 \notin C$ then $v_3$ is not connected to either $v_1$ or $v_2$ by any path.*

Components are an important concept in the study of community detection. Recall the intuition for a community introduced in section 1. From here, it is clear to see a similarity between the notion of a community and that of a component. Loosely put, a community is a subset of a network that is *nearly* a component.

## 2.5 Cut Sets

Cut sets, as the name would suggest, are sets of vertices that, if removed, cut the network into multiple components.

**Definition 9.** *(Cut Set) A cut set is a subset, $\mathcal{C}$, of the edge set, $E$, such that the network $N = (V, E \setminus \mathcal{C})$ has more than one component.*

(a) A strongly connected network (b) A loosely connected network

Figure 5: Two Erdős-Rényi [6] networks with differing connectedness rendered using NetworkX [9].

We also have the notion of a minimum cut set which is a cut set of minimum cardinality, i.e. it's the smallest subset of the vertices that can be removed which will disconnect the network into two components. Similarly to components, minimum cut sets are also important in the analysis of communities as the size of the minimum cut gives us some notion of how strongly connected our network is. A larger cut set means that we require more edges to be removed from the network to get multiple connected components. This suggests that a network with a larger minimum cut is more strongly connected and a network with a smaller minimum cut is more loosely connected. An example of such structure can be seen in Figure 5.

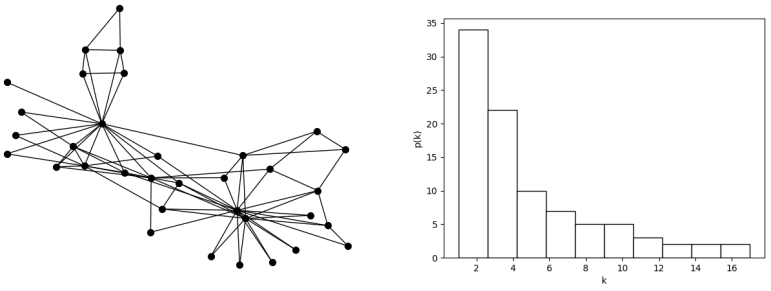
To make a link to computer science, we further introduce the ideas of *partitioning* and *clustering*. Clustering is a typical problem in computer science where one is presented with a graph and one wishes to break the graph into $N$ components using an appropriate cut set. The appropriate cut set is usually desired to be minimal. Partitioning and clustering is the formalisation of community detection where we take a network and figure out where the almost-components are via finding a smallest cut. This is discussed in more detail in Section 3.2.

## 2.6 Degree Distribution

We already know that the degree of a node is the number of edges incident on that node. In the context of adjacency matrices, the degree of node $i$ is defined by

$$\deg(i) = d_i = \sum_{j=1}^{N} A_{ij}.$$

There also exist similar definitions for both weighted and directed networks. Using this definition of the degree of a matrix, we can define something called the *degree distribution*. The degree distribution, as the name suggests, is a frequency distribution of all the degrees in the network. This distribution is often denoted by the function $p(k)$. This function counts the total number of




(a) The Zachary Karate Club (b) Degree Distribution

Figure 6: A network and its associated degree distribution plotted using NetworkX [9] and Matplotlib [11].

nodes with degrees greater than or equal to $k$. See, for example, Figure 6. Image 6a is a rendering of the Zachary Karate club and 6b is the associated degree distribution.

According to Renaud Lambiotte, these degree distributions often have very long tails which are regularly described by a power law:

$$p(k) \propto k^{-\gamma},$$

where $\gamma$ usually takes values between 2 and 3 [15, 16]. The relationship above holds approximately until some "cutoff degree" where the structure changes and $p(k)$ quickly decreases to 0. Networks that have a degree distribution of this form are commonly called scale-free and many models of real world networks end up having this property. For example, the internet [23], world wide web [2], metabolic networks [12] and citation networks [22] all appear to be scale-free.

## 2.7 Clustering Coefficient

When thinking about community detection, we're actually interested in the connectedness of different parts of the network and in particular we're interested in the interconnectedness of a set of vertices. One way to measure the interconnectedness of a vertex with the surrounding nodes is using the clustering coefficient. The clustering coefficient counts the number of triangles in the network that include a given vertex. We define the clustering coefficient in the following way

$$C_i = \frac{\text{number of triangles including the } i\text{th node}}{k_i(k_i - 1)/2}.$$

This quantity measures and normalises the number of triangles in the immediate vicinity of the vertex $i$. Using the clustering coefficient, we can extend this to consider the whole network:

$$C = \frac{1}{N} \sum_{i=1}^{N} C_i.$$

Again this total measure of clustering is normalised such that $0 \leq C \leq 1$.

## 2.8 Centrality

The core idea of centrality is to formalise the importance of certain nodes in the network. There are, of course, a number of ways to do this. Lambiotte covers a number of centrality measures [15, 18] which are reprinted here in order of increasing complexity. To begin, we shall talk about closeness centrality. Simply put, the closeness centrality is the inverse of the mean distance between a node $i$ and every other node in the network. Notationally, this looks like the following:

$$\text{closeness}_i = \frac{N-1}{\sum_{j=1;j\neq i}^{N} d(i,j)},$$

where $d(i,j)$ is the length of a geodesic path between $i$ and $j$. As an initial measure of centrality, this holds up well. It clearly meets the criterion of identifying nodes that are important in the network as nodes that are close to most other nodes have a very high closeness centrality whilst nodes that exist on the peripherals of a network have a low closeness centrality.

However, this measure doesn't consider the importance of nodes when traversing the network. Betweeness centrality looks to solve this problem by counting the each node's contribution to the number of shortest paths that exist in the network. Betweeness centrality is calculated in the following way:

$$\text{betweenness}_i = \frac{2}{(N-1)(N-2)} \sum_{j=1;j\neq i}^{N} \sum_{l=1;l\neq i}^{j-1} \frac{\sigma_{jl}^{i}}{\sigma_{jl}},$$

where $\sigma_{jl}$ denotes the total number of shortest paths connecting nodes $j$ and $l$ and $\sigma_{jl}^{i}$ is the number of such paths containing the node $i$. This measure clearly fixes the issue that closeness centrality had by considering how many paths use a given node $i$, but what if a node rarely contributes to shortest paths, but often contributes to near-shortest paths? We would expect a good measure of the importance of a node in a network to use at least some contribution from near shortest paths. Katz centrality does this by considering all paths connecting a node to every other node and giving longer paths less weight. Katz centrality is calculated as follows:

$$\text{Katz}_j = \sum_{i=1}^{N} [(I - \alpha A)^{-1}]_{ij},$$

where $\alpha \in (0,1)$ is an absolute constant. These methods all build up to more complicated measures such as PageRank [20] which solves exactly the same problem, but using more advanced and intricate techniques and not necessarily on undirected networks.

## 2.9 Spectral Properties

The final properties of interest are spectral in nature. Spectral properties are properties that are based on the eigendecomposition of either the adjacency matrix, the Laplacian or a modified version of the Laplacian called the normalised

Laplacian. If the Laplacian is defined by

$$L = D - A,$$

then the normalised Laplacian is given by

$$\tilde{L} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} A D^{-1/2}.$$

By definition, the adjacency matrix, the Laplacian and the normalised Laplacian are symmetric (for an undirected network). This means that the eigenvalues of each matrix are all real and the corresponding eigenvectors form an orthonormal basis. It's important to note that the two Laplacian matrices always have $\lambda_1 = 0$. In fact, if the network is connected, $\lambda_1 = 0$ and $\lambda_i > 0 \; \forall i > 1$. This is our first spectral property of a matrix. More advanced spectral properties are used to solve community detection problems such as finding a minimum cut (see section 3.2). Introduction of spectral ideas concludes this section on properties of interest. Now that all of the necessary vocabulary and concepts are introduced, we are ready to discuss community detection.

# 3 Community Detection

As alluded to in the previous sections, detecting communities is of great interest and as such there a number of ways to do it. The process of community detection involves analysing the network and finding groups of nodes in the network that are more densely connected amongst themselves than they are to the rest of the network. This notion forms the basis for most community detection methods and algorithms. However, a robust definition of community is difficult to formulate. In fact, it turns out that "in most cases, communities are algorithmically defined, i.e. they are just the final product of the algorithm without a precise *a priori* definition." [8, 84] In this section, we will discuss the notions of "community" that underpin a number of interesting methods in community detection before going into detail about the algorithms themselves.

A simple example of the aforementioned notion comes in the form of inter and intra-cluster densities as discussed by Fortunato [8, 84]. This idea assumes that we have a network $N$ and a subnetwork $C \subseteq N$ i.e. $C$ is also a network where every node and edge in $C$ is also in $N$, but the converse is not necessarily true. Suppose we want to determine whether $C$ is a community inside $N$. To aid us in our investigation, we can define the following two quantities,

1. Intra-cluster density: $\delta_{\text{int}}(C) = \frac{\text{\# of internal edges of C}}{n_c(n_c-1)/2}$,
2. Inter-cluster density: $\delta_{\text{ext}}(C) = \frac{\text{\# of external edges of C}}{n_c(n_c-1)/2}$,

where $n_c = |C|$ and internal and external edges refer to edges originating in $C$ that end inside and outside $C$ respectively. Thus the intra-cluster density is the ratio of edges originating in $C$ that remain in $C$ whilst the inter-cluster density is the ratio of edges originating in $C$ that end outside $C$. Finally, to make sense of these metrics, we introduce a fiducial marker for community structure, the average link density:

$$\delta(N) = \frac{\text{\# of edges in } N}{n(n-1)/2}.$$

Following the same intuition as before, this quantity represents the ratio of edges that are present in the network to the total number of possible edges. Now, if $C$ were a community, we would expect that $\delta_{\text{int}}(C)$ is noticeably larger than $\delta(N)$. Similarly, we would expect $\delta_{\text{ext}}(C)$ to be noticeably smaller than $\delta(N)$. Getting this result is the goal of most community detection algorithms.

Stricter definitions of communities come in three flavours:

1. Local definitions which rely on considering the structure of a given subnetwork and perhaps it's immediately adjacent neighbours,
2. Global definitions which consider the structure of the whole network,
3. Vertex similarity definitions which rely on the notion of similarity between any two vertices.

Each method seeks to formalise our intuition that communities should be strongly connected amongst themselves, but weakly connected to the rest of the network. To extend this intuition, we will refer to Wasserman's summary

of the four general properties of what he calls "cohesive subgroups" (but we call communities) that have influenced the formalisations and definitions of the concept in the social network literature [25, 251-252]:

1. The mutuality of ties (cliques);
2. The closeness or reachability of subgroup (community) members;
3. The frequency of ties amongst members (frequency of connections between community members);
4. The relative frequency of ties amongst subgroup (community) members compared to non-subgroup (non-community) members.

Each of these points refers to a different strategy for identifying communities. Paraphrasing Wasserman and using our terminology: Strategies based on cliques require each member of a community to be directly adjacent to each other member of a community; strategies based on closeness or reachability require that each member of a community is reachable from every other member of the same community, but adjacency is not required; strategies based on frequency of connetions between community members require that each member of the community is adjacent to many other members of the community; and strategies based on the relative frequency of connections require that members of the community are more connected amongst each other than they are to the rest of the network.

These four ideas for definitions are such that the communities they generate are considered maximal subnetworks i.e. adding another node to the subnetwork would remove the subnetwork's community property. The ideas are also in decreasing order in terms of strictness and Fortunato provides an excellent summary of the journey from a rudimentary strategy using cliques to strategies involving fitness measures in the case of local definitions. cite[88-90]fortunato

Global definitions, of course, follow the same notion, but they have a different motivation. Whilst local definitions are used when we're interested in just the structure of the subnetwork, global definitions are used when the community structure doesn't make sense without the context of the rest of the network and vice versa. Global definitions are more often indirect definitions in the sense that the communities are defined by the output of an algorithm implementing a global method. Curiously though, there are a set of direct global definitions. These definitions are called *null model* definitions. These definitions rely on taking the original network and generating a random version of it that has some similarity in it's structural features to the original. These two networks are then compared in some way which will reveal any underlying community structure. In fact, these null model methods underpin the idea of modularity; a concept which is relied upon by the most popular method of network clustering.

## 3.1 Quality Functions and Basic Modularity

While considering certain dichotomies or methodological aspects is a meaningful first step, we are ultimately interested in the efficacy of a given method. Anyone can define an algorithm that breaks a network down into communities (for example by putting each node into its own community), but some of these might

not provide us with high quality insight to the structure of a network. Hence we develop the idea of *quality functions*. A community detection algorihm takes a network $N$ and returns a set of partitions and/or clusters in that network based on a set of rules. A quality function determines the quality of an algorithm by assigning a number to each partition of a network. Typically, partitions with a high score are considered "good". According to Fortunato, the most used quality function is Newman and Girvan's definition of modularity[19, 8]:

$$Q = \sum_i (e_{ii} - a_i^2) = \mathrm{Tr}(e) - ||e^2||.$$

This definition relies on constructing a matrix $e$ such that $e_{ij}$ is the fraction of edges in the network that link a node in community $i$ to a node in community $j$, $a_i = \sum_j e_{ij}$ the fraction of edges that connect to vertices in community $i$, and finally $||e^2||$ denotes the sum of all elements of the matrix $e^2$. Based on this definition, if our algorithm has partitioned the network successfully into communities then we expect $Q$ to be large. Even though $\mathrm{Tr}(e)$ is large when the communities are very well connected, it achieves a maximum of $\mathrm{Tr}(e) = 1$ when all the nodes are in the same community. This is clearly not useful as it doesn't give us any information about the structure of the networks. Thus we subtract $||e^2||$ from $\mathrm{Tr}(e)$. To quote Newman and Girvan: "this quantity measures the fraction of edges in the network that connect vertices of the same type minus the expected value of the same quantity in a network with the same community divisions but with random connections between them". The idea for this definition is to use a *null model* and exploit the fact that random networks are not expected to have any community strucutre.

## 3.2 Community Detection via Spectral Smallest Cut

The simplest way to consider community detection is to think about dividing the network into subnetworks such that the number of edges between all the subnetworks is minimised. This is called a smallest cut. Referring again to Lambiotte's notes, we will define a few pieces of machinery that will allow us to algorithmically find the set of edges that separates the communities [15, 26-27]. For simplicity, we restrict ourselves to trying to identify only two communities. Let us assume that we have partitioned the graph into two groups labelled 1 and 2. We then define the number of edges starting in group 1 and ending in group 2 or vice versa by the following quantity:

$$R = \frac{1}{2} \sum_{\substack{i,j \text{ in} \\ \text{different} \\ \text{groups}}} A_{ij}.$$

Of course, this isn't very easy to work with, so we define a vector $s$ such that

$$s_i = \begin{cases} +1 & \text{if vertex } i \text{ belongs to group 1,} \\ -1 & \text{if vertex } i \text{ belongs to group 2.} \end{cases}$$

After noting the following

$$\frac{1}{2}(1 - s_i s_j) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are in different groups,} \\ 0 & \text{if } i \text{ and } j \text{ are in the same group,} \end{cases}$$

we can rewrite $R$ as

$$\begin{aligned} R =& \frac{1}{4}\sum_{ij}(1 - s_i s_j)A_{ij} \\ & \vdots \\ =& \frac{1}{4}\sum_{ij} s_i s_j (k_i \delta_{ij} - A_{ij}), \end{aligned}$$

which is then equal to the following,

$$R = \frac{1}{4} s^T L s,$$

where $L$ is the Laplacian matrix of the network. So now finding the minimum cut is equivalent to finding the vector $s$ that minimises the above quantity. The trick is to rewrite $s$ using a linear combination of the normalised eigenvectors, $v_i$ of $L$. This means that $s$ takes the form $\sum_{i=1}^{n} a_i v_i$ where we set $a_i = v_i^T s$. It should also be simple to see that $s^T s = n$. Combining this redefinition of $s$ and the $a_i$s, we can note that

$$\sum_{i=1}^{n} a_i^2 = n.$$

Letting $\lambda_i$ be the eigenvalue of $L$ correspnding to $v_i$, we get the following

$$R = \sum_i a_i v_i^T L \sum_j a_j v_j = \sum_{ij} a_i a_j \lambda_j \delta ij = \sum i\lambda_i.$$

Convention dictates that we label the eigenvalues in increasing order, i.e. $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. In order to minimise the value of $R$, we wish to chose an $s$ that places as much weight on the smaller eigenvalues as possible. With this in mind, the easiest way to minimise $R$ is to choose $s = (1, 1, \dots) = v_i$ the eigenvector corresponding to $\lambda_i = \lambda_1 = 0$. This, however, is not an interesting solution as all it does is put every node into group 1 and none into group 2. To fix this, we prohibit this solution and find the next best one. The next best solution places as much weight as possible on the second eigenvalue $\lambda_2$ of the Laplacian. To do this, we select an $s$ that is proportional to the second eigenvector, $v_2$. This eigenvector is commonly referred to the *Fiedler vector*. However, due to the constraint that every $s_i = \pm 1$ we can rarely chose an $s$ that is exactly proportional to $v_2$. Lambiotte provides a more comprehensive discussion of this argument, but in this case it is often sufficient to chose $s$ as close to $v_2$ as possible.

| | Karate | Arxiv | Internet | Web nd.edu | Phone | Web uk-2005 | Web WebBase 2001 |
|---|---|---|---|---|---|---|---|
| Nodes/links | 34/77 | 9k/24k | 70k/351k | 325k/1M | 2.6M/6.3M | 39M/783M | 118M/1B |
| CNM | .38/0s | .772/3.6s | .692/799s | .927/5034s | -/- | -/- | -/- |
| PL | .42/0s | .757/3.3s | .729/575s | .895/6666s | -/- | -/- | -/- |
| WT | .42/0s | .761/0.7s | .667/62s | .898/248s | .56/464s | -/- | -/- |
| Our algorithm | .42/0s | .813/0s | .781/1s | .935/3s | .769/134s | .979/738s | .984/152mn |

Figure 7: A comparison of the Louvain method (noted in the table as "Our algorithm") against methods from Clauset, Newman and Moore (CNM) [4], Pons and Latapy (PL) [21] and Wakita and Tsurumi (WT) [24]. The first row shows the number of nodes/links in each data set whilst all other rows show the modularity of the result from the algorithm and the time taken to compute the result. The symbols "-/-" are for combinations of algorithm and dataset where the computation time was greater than 24 hours. This table is taken from [3].

After some more algebra and manipulation, it transpires that the optimal choice for any $s_i$ under this paradigm is

$$s_i = \begin{cases} +1 & \text{if } v_i^{(2)} \geq 0, \\ -1 & \text{if } v_i^{(2)} < 0, \end{cases}$$

where $v_i^{(2)}$ is the $i$-th element of $v_2$. There are a few more minor issues involving constraints that are not discussed here that Lambiotte covers in good detail [15, 27]. This is perhaps the simplest method of community detection, though it is limited in its scope and power due to the fact that it can only separate the network into two groups when in reality we might be interested in $N \gg 2$ groups.

### 3.3 Louvain Community Detection

To remedy the aforementioned issue and to introduce some more recent developments in the field, we will now explore a more advanced method known as Louvain detection [3], named after the University of Louvain in Belgium. Louvain detection is *extremely* fast and *extremely* effective (see Figure 7) and operates on weighted networks as with the Ford-Fulkerson method (Section 4.1). Figure 7 shows a comparison of the Louvain method against other contemporary methods. As you can clearly see, Louvain community detection is significantly faster than all other compared methods taking only 152 minutes to compute the community structure of a network with 118 million nodes and 1 billion edges. Part of the Louvain method's efficiency is that it's broken into two phases:

1. Modularity optimisation;
2. Community aggregation.

To set up the method, we take every node in the network and put it into it's own community i.e. if there are $N$ nodes in the network then there are $N$ communities. Once we have done this, we begin phase one.

In phase one we iterate over every node, $i$, in the network and consider it's neighbours, $j$. Then, for each $j$, we consider the change in modularity of the

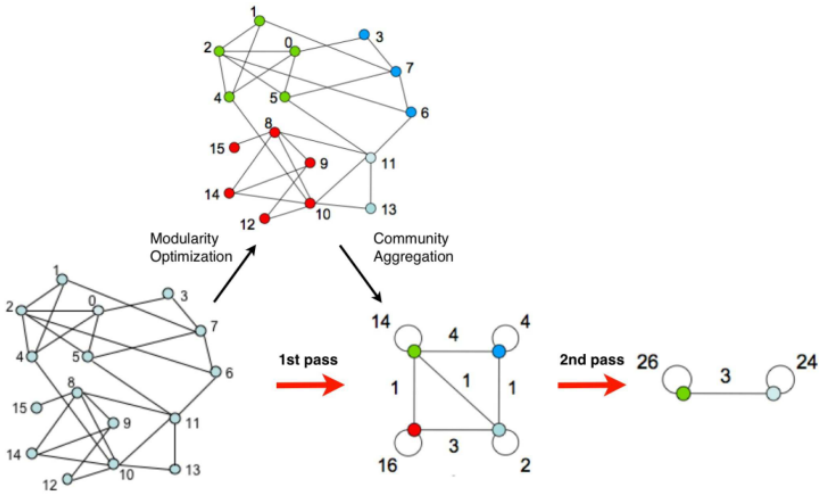



Figure 8: A diagram showing multiple applications of phases one and two (passes) of the Louvain algorithm from [3].

network by adding $i$ to the community of $j$. We then add the node $i$ to the community of $j$ that maximises the increase in modularity. If no movement of a node $i$ into the community of $j$ increases the modularity, we leave $i$ in its current community and proceed onto the next node. This process is then repeated until no single move of a vertex $i$ into the same community as $j$ increases the modularity.

We then begin phase two. Phase two consists of taking all of the communities found in phase one and considering them to be nodes in a network. In this network, the links between nodes are given by the sum of the weights of edges between the respective communities and a self-link is added to every node with a weight equal to the sum of the weights of the links inside the community. We call the application of a phase one followed by a phase two a "pass" and a diagram of this process can be seen in Figure 8.

As mentioned previously, the Louvain method is extremely efficient, but where does this efficiency come from? The answer to this question is twofold. On one front, the recursive nature of the algorithm significantly reduces the size of the problem very quickly meaning that after the first pass, even for lager data sets, it can be very quick to find solutions. Secondly, it turns out that after some algebraic manipulation, it's very computationally cheap to calculate the change in modularity for any given move from $i$ into the community $C$:

$$\Delta Q = \left[\frac{\Sigma_{\text{in}} + k_{i\text{in}}}{2m} - \left(\frac{\Sigma_{\text{tot}} + k_i}{2m}\right)^2\right] - \left[\frac{\Sigma_{\text{in}}}{2m} - \left(\frac{\Sigma_{\text{tot}}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right],$$

where $\Sigma_{\text{in}}$ is the sum of the weights of the links inside $C$, $\Sigma_{\text{tot}}$ is the sum of the weights of the lnks incident to nodes in $C$, $k_i$ is the sum o fthe weights of the links incident to the node $i$ $k_{i,\text{in}}$ is the sum of the weights of the links from $i$ to nodes in $C$ and $m$ is the sum of all the weights of all links in the network.

Louvian detection also uses a slightly different definition of modularity that is also given by Newman[18]:

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

where $A$ is the weighted adjacency matrix for the network, $\delta$ is the Kronecker delta and $c_i$ is the community to which vertex $i$ is assigned. These two qualities, when combined, result in the extrodinary efficiency shown in Figure 7.

These are only two examples and are the two extremes of algorithms in the field; one very rudimentary algorithm and one very complex algorithm. Other methods exist from a variety of authors such as Clauset, Newman and Moore [4], Wakita and Tsurumi [24] and Pons and Latapy [21] and each method has its own advantages and disadvantages. Now that examples of methods have been established, we move on to considering applications of these ideas.

# 4 Applications of Community Detection

We will start with the simple example of Zachary's Karate Club to illustrate the key concepts before the methodologies explored are deployed in a more extensive context where it is difficult to discern certain features as cleanly.

## 4.1 Zachary's Karate Club

To add more to the context already introduced in Section 1.1, the karate club that Zachary studied had two main leaders who are referred to in the paper as "Mr Hi" and "John A". Due to some inter-personal politics, members of the club ended up becoming part of a faction that was politically aligned with one of the primary leaders. After a period of in-fighting, the club split into two distinct groups. Zachary collected all the data relating to this including information on how often any two members of the group interacted, which leader they were factionally affiliated with and which group each member was a part of post-fission. Using this data, Zachary built a model that knew how much any two particular members interacted, but had no knowledge of their factional affiliations. After developing this model, Zachary was interested in the following two hypotheses:

1. Can we predict the political affiliation of any member of the club pre-fission?
2. Can we predict which club any member will join after the split?

Zachary chose to model this problem as a network flow problem so that he could use the *max-flow min-cut theorem* and the *Ford-Fulkerson algorithm* [7] to test his hypotheses. The max-flow min-cut theorem states that the maximum flow across a network is equal to the capacity of the minimum cut and the Ford-Fulkerson algorithm is a deterministic method for finding the minimum cut. Using this model we can rephrase the above two hypotheses:

1. The minimum cut of the network should separate those factionally affiliated with Mr Hi from those factionally affiliated with John A.
2. The minimum cut of the network should separate those who choose to join Mr Hi's club after the split from those who choose to join John A's club after the split.

After running the Ford-Fulkerson algorithm on the data, Zachary achieved results that correctly predicted 34 out of 34 faction memberships and 33 out of 34 club memberships. The notable exception being individual number 9 who reportedly joined Mr Hi's club so as not to miss out on a black belt. Figure 9 shows Zachary's full results.

This is a prime example of the predictive power of community detection: knowing nothing about the inter-group politics, Zachary was able to *very* accurately predict the way the club interacted and split after a dispute.

**EVALUATION OF THE HYPOTHESES**

| INDIVIDUAL NUMBER IN MATRIX *C* | FACTION MEMBERSHIP FROM DATA | FACTION MEMBERSHIP AS MODELED | HIT/ MISS | CLUB AFTER SPLIT FROM DATA | CLUB AFTER SPLIT AS MODELED | HIT/ MISS |
|---|---|---|---|---|---|---|
| 1 | Mr. Hi | Mr. Hi | Hit | Mr. Hi's | Mr. Hi's | Hit |
| 2 | Mr. Hi | Mr. Hi | Hit | Mr. Hi's | Mr. Hi's | Hit |
| 3 | Mr. Hi | Mr. Hi | Hit | Mr. Hi's | Mr. Hi's | Hit |
| 4 | Mr. Hi | Mr. Hi | Hit | Mr. Hi's | Mr. Hi's | Hit |
| 5 | Mr. Hi | Mr. Hi | Hit | Mr. Hi's | Mr. Hi's | Hit |
| 6 | Mr. Hi | Mr. Hi | Hit | Mr. Hi's | Mr. Hi's | Hit |
| 7 | Mr. Hi | Mr. Hi | Hit | Mr. Hi's | Mr. Hi's | Hit |
| 8 | Mr. Hi | Mr. Hi | Hit | Mr. Hi's | Mr. Hi's | Hit |
| 9 | John | John | Hit | Mr. Hi's | Officers' | Miss |
| 10 | John | John | Hit | Officers' | Officers' | Hit |
| 11 | Mr. Hi | Mr. Hi | Hit | Mr. Hi's | Mr. Hi's | Hit |
| 12 | Mr. Hi | Mr. Hi | Hit | Mr. Hi's | Mr. Hi's | Hit |
| 13 | Mr. Hi | Mr. Hi | Hit | Mr. Hi's | Mr. Hi's | Hit |
| 14 | Mr. Hi | Mr. Hi | Hit | Mr. Hi's | Mr. Hi's | Hit |
| 15 | John | John | Hit | Officers' | Officers' | Hit |
| 16 | John | John | Hit | Officers' | Officers' | Hit |
| 17 | Mr. Hi | Mr. Hi | Hit | Mr. Hi's | Mr. Hi's | Hit |
| 18 | Mr. Hi | Mr. Hi | Hit | Mr. Hi's | Mr. Hi's | Hit |
| 19 | John | John | Hit | Officers' | Officers' | Hit |
| 20 | Mr. Hi | Mr. Hi | Hit | Mr. Hi's | Mr. Hi's | Hit |
| 21 | John | John | Hit | Officers' | Officers' | Hit |
| 22 | Mr. Hi | Mr. Hi | Hit | Mr. Hi's | Mr. Hi's | Hit |
| 23 | John | John | Hit | Officers' | Officers' | Hit |
| 24 | John | John | Hit | Officers' | Officers' | Hit |
| 25 | John | John | Hit | Officers' | Officers' | Hit |
| 26 | John | John | Hit | Officers' | Officers' | Hit |
| 27 | John | John | Hit | Officers' | Officers' | Hit |
| 28 | John | John | Hit | Officers' | Officers' | Hit |
| 29 | John | John | Hit | Officers' | Officers' | Hit |
| 30 | John | John | Hit | Officers' | Officers' | Hit |
| 31 | John | John | Hit | Officers' | Officers' | Hit |
| 32 | John | John | Hit | Officers' | Officers' | Hit |
| 33 | John | John | Hit | Officers' | Officers' | Hit |
| 34 | John | John | Hit | Officers' | Officers' | Hit |
| TOTALS | | 34 hits, 0 misses<br>100% hits, 0% misses | | | 33 hits, 1 miss<br>97% hits, 3% misses | |

Figure 9: The full results from Zachary's investigation into using the Ford-Fulkerson minimum-cut algorithm to predict social and political alignments [26].

## 4.2 Epidemic Spreading and Community Structure

From experience, we understand that community structure might make a difference to the spreading of a virus or disease through a population.[1]. Until recently, the idea of analysing the difference in epidemic propagation between networks with and without communities remained nearly untouched. A paper by Huang and Li published in 2007 [10] investigated the difference between epidemic spreading in scale-free networks with and without community structure. Recall from Section 2.6 that scale free networks have a degree distribution that rougly follows a power law. Further recall that scale-free networks are interesting because they represent a plethora of real world complex systems.

Huang and Li's method involves creating a number of realisations of scale-free networks (SFNs) and scale-free networks with communities (SFcNs) and performing Monte Carlo simulations for each realisation using a Susceptible-Infected (or SI) model. This model works by labelling each member of the network either susceptible, where the member can be infected by the disease, or infected, where the member currently has the disease and can infect others. The probability with which any infected member does this is parameterised by the authors using $\lambda$.

To generate the SFcNs, Huang and Li refer to a paper by previous paper by Li et. al. [16] that provides a method for generating scale-free networks with a community structure. The method is as follows:

1. Initialisation: Choose $M \geq 2$ as the number of communities and $m_0 > 1$ the number of fully connected nodes in each community. Then add a link between every pair of communities so that there are a total of $M(M-1)/2$ inter-community links. These links are uniformly randomly assigned to nodes inside each community.

2. Growth: In each iteration, we add a new node to a uniformly randomly selected community. This new node will be uniformly randomly connected to $1 \leq m \leq m_0$ nodes in the same community and with a probability $\alpha$ it will be connected to $1 \leq n \leq M$ nodes in the other $M-1$ communities.

3. Preferential attachments

   (a) Intra-community attachments: The probability of a new node being connected to a node $i$ in community $j$ is depends on the inner-degree $s_{ij}$ which is the number of intra-community links connected to $i$. The dependence is as follows:

   $$\Pi(s_{ij}) = \frac{s_{ij}}{\sum_k s_{kj}}.$$

   (b) Inter-community attachments: Similarly to before, the probability of a new node being connected to a node $i$ in community $k$ depends on the inter-degree $l_{ik}$ which is the number of inter-community links

[1] This wouldn't be an essay written in the 2020s without mentioning epidemic processes!

connected to node $i$. This dependence is as follows:

$$\Pi(l_{ik}) = \frac{l_{ik}}{\sum_{\substack{m,n \\ n \neq j}} l_{mn}}.$$

When the above is repeated for enough iterations, we get at SFcN with $N$ nodes and $M$ communities. To understand how strong the community structure generated by this method is, Huang and Li use the definition of modularity proposed by Newman and Girvan (see Section 3.1). Using this measure of modularity and some results from [16] we can obtain the modularity in terms of the parameters for the algorithm:

$$Q = \frac{m}{m + \alpha n} - \frac{1}{M}\left(\frac{m + 2\alpha n}{m + \alpha n}\right)^2.$$

Thus, the authors fix the values of $m$ and $n$ and adjust the values of $\alpha$ to get networks with various strengths of community structure. Of course, for a fair test, the authors need a SFN with precisely the same degree distribution as the SFcN. Thus they perform a series of *Monte Carlo vertex switching* steps on the SFcN to generate such a SFN. A single switching step involves taking a pair of links $\{A, B\}$ and $\{C, D\}$ and switching the ends to get $\{A, D\}$ and $\{C, B\}$. This switch is only performed if it doesn't form any multiple or self links and clearly preserves the degree distribution. To ensure that there's enough mixing, the authors switch a total of $5M$ pairs of links. After all this set up, the authors then run the aforementioned Monte Carlo simulations for 30 different values of $\alpha$ and for each value of $\alpha$ they simulated 500 different initial conditions with one node chosen at random to be infected. The results from these simulations show that community structure has an impact on the spreading of the disease. Figure 10 shows the results for $\alpha = 0.01$ which gives a modularity $Q \approx 0.845$. In order from top left to bottom right, the plots represent the following things:

- the proportion of individuals that are infected at time $t$,
- the variability of the proporotion of individuals that are infected at time $t$,
- the average degree of infected nodes at time $t$,
- the variability of the average degree of infected nodes at time $t$.

Observing the graphs we can see that SFNs behave differently to SFcNs. Most interestingly we can see that the proportion of infected individuals is *lower* for SFcNs than it is for SFNs at any time $t$ and that the variability of the proportion of infected individuals peaks lower. This is significant because it informs our intuition that separating parts of the network results in slower transmission and lower infection proportions. Results like this are important because they have implications about how we adapt and inform our actions when faced with real world problems.

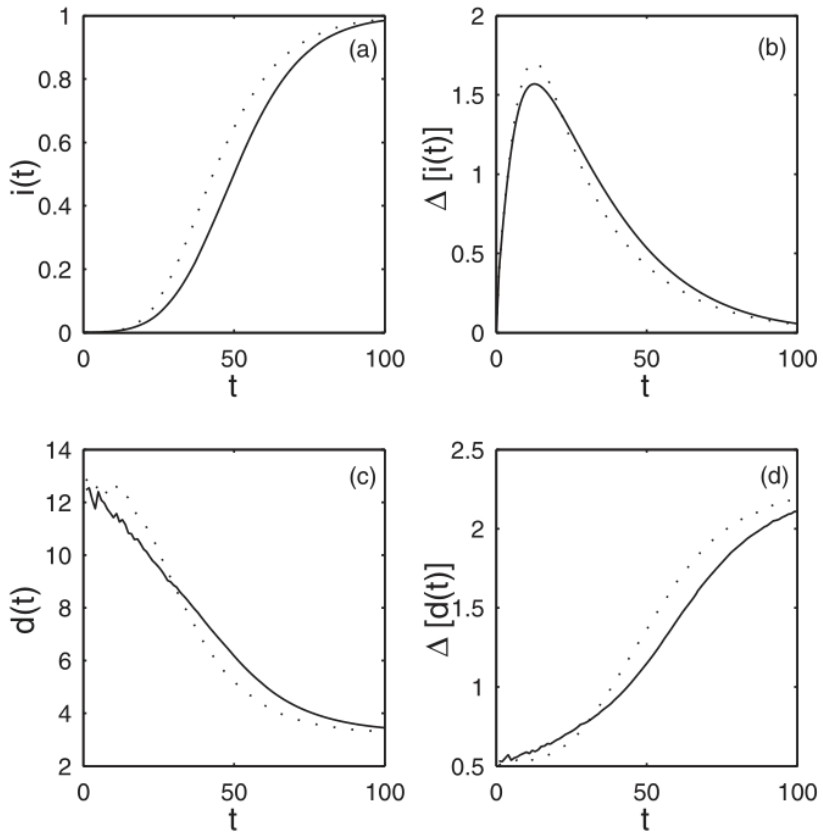



Figure 10: Results from Huang and Li's analysis of the difference in epidemic processes on SFNs and SFcNs for $\alpha = 0.01$, $\lambda = 0.03$ and $N = 2000$. SFNs are represented by the dotted line and SFcNs are represented by the solid line.

# 5 Conclusion

## 5.1 Overview

As a result of the topic being very computational in nature, community detection has scaled with computational power and this means that it is quite a young field. Despite this, a huge amount of research has been done on the topic thus far and Fortunato has done amazing work collating a large amount of the available information [8]. However, as always, there still remain open problems in the field. Recall Section 3: In this section, we discussed that currently there is no strict definition of a community and that we typically define communities by the results of community detection algorithms and methods rather than having a definiton a priori. This appears to be the primary open problem in the field right now. Coming up with a strong and well supported definition of a community would let us better understand the quality of our methods and strategies for finding them. Failing this, the next best thing would be to design a comprehensive set of networks and suite of tests that we can give to community detection algorithms to test their efficacy. These networks and tests should obviously cover as many edge cases as possible to ensure that our algorithms work even for the most difficult to detect communities.

It should further be noted that community detection and the ideas of community work both ways. Recall Section 4. In this section we discussed two applications of community detection: The Zachary Karate Club and Epidemic Spreading and Community Structure. In the first example, we are taking a given network and trying to analyse the underlying community structure, but in the second example we start with a network and show that by creating a community structure we gain some favourable characteristics. These two qualities of community detection when paired together make a very powerful framework and paradigm for investigating real world problems.

## 5.2 Personal Learnings

Throughout the course of writing this essay, I have developed a strong understanding of the foundations of research into community detection. What was most interesting to me was that there is no single strict definition of a community and instead we choose to define communities by the results of algorithms (Section 3). This was a shock to me, but also unsurprising. A shock because I had expected that, given the utility and power of community detection in understanding real world phenomena, we would have a solid way to define what it is we're talking about. Unsurprising because networks are intuitively extremely complex on a large scale and, as such, identifying macroscopic properties such as community structure is understandably very hard. The most important personal learning from the writing of this essay was about quality functions and modularity. Of course it is required that there is some quantitative measure of the performance of an algorithm, but until researching them I had underappreciated the intricacy required to make such a measure. Take, for example, the Newman-Girvan modularity from Section 3.1. This measure is not as simple as "calculate the fraction of edges that go between communities". Instead there is a very deliberate design process and set of criteria that a quality function should meet such as normality (the property of being normalised) as well as the ability

to omit trivial identifications of a community (such as putting all the nodes in the same community - this is technically a perfect detection of a community, but it doesn't help us understand the structure of the network in any way).

It should now be clear that networks and community detection are of great interest to the scientific community for their applications in modelling real world complex systems such as epidemic processes (Section 4.2) and social dynamics (Section 4.1). Most importantly of all, we appreciate that community detection may be useful whenever you have a system that can be modelled as a network. For example: understanding the topology of the world wide web [2] or how metabolic networks are organised [12]. All of this paired with the fact that our ability to solve large computational problems is ever increasing make communities a topic of interest for the future. I hope that the reader finds themselves with enough information to consider community structure when working on their next problem.

# References


[1] Zachary karate club network dataset – KONECT, Oct. 2017.

[2] A.-L. Barabási, R. Albert, and H. Jeong. Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A: Statistical Mechanics and its Applications*, 281(1):69–77, 2000.

[3] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, Oct 2008.

[4] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(6), Dec 2004.

[5] A. Davis, B. B. Gardner, and M. R. Gardner. *Deep South; a Social Anthropological Study of Caste and Class.* The Univ. of Chicago Press, 1941.

[6] P. Erdös and A. Rényi. On random graphs I. *Publicationes Mathematicae Debrecen*, 6:290, 1959.

[7] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[8] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[9] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught, and J. Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[10] W. Huang and C. Li. Epidemic spreading in scale-free networks with community structure. *Journal of Statistical Mechanics: Theory and Experiment*, 2007(01):P01014–P01014, Jan 2007.

[11] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[12] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, Oct 2000.

[13] S. G. Kobourov. Spring embedders and force directed graphs. 2012.

[14] J. Kunegis. KONECT – The Koblenz Network Collection. In *Proc. Int. Conf. on World Wide Web Companion*, pages 1343–1350, 2013.

[15] R. Lambiotte. C5.4 networks lecture notes. 2021.

[16] C. Li and P. K. Maini. An evolving network model with community structure. *Journal of Physics A: Mathematical and General*, 38(45):9741–9749, Oct 2005.

[17] M. Newman. *Networks: An Introduction.* Oxford University Press, 2010.

[18] M. E. J. Newman. Analysis of weighted networks. *Phys. Rev. E*, 70:056131, Nov 2004.

[19] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, Feb 2004.

[20] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, Nov 1999. Previous number = SIDL-WP-1999-0120.

[21] P. Pons and M. Latapy. Computing communities in large networks using random walks. *Journal of Graph Algorithms and Applications*, 10(2):191–218, 2006.

[22] S. Redner. How popular is your paper? an empirical study of the citation distribution. *The European Physical Journal B - Condensed Matter and Complex Systems*, 4(2):131–134, Jul 1998.

[23] A. Vázquez, R. Pastor-Satorras, and A. Vespignani. Large-scale topological and dynamical properties of the internet. *Phys. Rev. E*, 65:066130, Jun 2002.

[24] K. Wakita and T. Tsurumi. Finding community structure in mega-scale social networks, 2007.

[25] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Structural Analysis in the Social Sciences. Cambridge University Press, 1994.

[26] W. Zachary. An information flow model for conflict and fission in small groups. *J. of Anthropol. Res.*, 33:452–473, 1977.