Community Detection in Complex Networks

Joel McFarlane
CID: 01347129

Supervisor: Tim Evans
Assessor: Kim Christensen
Imperial College London

6th March 2020

Abstract: The project examined methods of disjoint community detection within undirected, unweighted networks. Three methods were investigated, Spectral Modularity Optimisation, the Louvain method, and the InfoMap method. Each of the methods was tested, first for accuracy on a network with known community structure, then for speed on a variety of networks of different sizes. The Louvain method was found to be the most accurate on the particular network tested, while the InfoMap method was the fastest on every network tested. Finally, the errors on the methods were investigated and applications of community detection were explored.

Word count: Approximately 5150 words in the report (excluding front page, figure captions, table captions, acknowledgement and bibliography).

Declaration: All research in the essay was carried out by the author. All figures and computational work were carried out by the author unless otherwise stated.

Contents

1 Introduction

Network Science aims to provide a representation of complex systems with many elements that interact with each other [1]. This approach has a wide variety of applications, from modeling traffic systems, social networks and neural networks [2]. Of course, due to the fact that Networks only provide a representation of a system, some information will be lost [3]. However, important conclusions can be drawn from the Network that otherwise wouldn't.

The aim of this essay will be to explore community detection on networks. The first part will outline some basic concepts and definitions that will be used later on. Following this, the methods to identify community structure of networks will be discussed and some of these methods will be applied to real-life networks. The methods discussed include both modularity maximization algorithms as well as non-modularity based algorithms. The final aim will be to explore some applications of community detection in networks.

2 Definitions and Representation

To start with, the basic features of a network are outlined. The network is defined as a set of nodes, that are connected to each other via edges. This network can be represented as a graph $G = (V, E)$, where V is a set of nodes and E a set of edges [4]. Figure 1 shows a basic example. This is a simple network, one where there is at most one edge between two nodes and no edges where both sides are connected to the same node. During this essay the focus will be on simple networks however, the methods discussed can be applied to other networks as well.

There are many ways to represent a network however, during this essay the Adjacency Matrix will be used. In this representation $A_{j,i} = 1$ if there is an edge between node V_i and node V_j, whereas $A_{j,i} = 0$ if there is no edge [5]. Equation (1) shows an example of an adjacency matrix, where here there is an edge between nodes 1 and 2, and an edge between nodes 1 and 3.

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}. \tag{1}$$

The advantage of using an adjacency matrix to store the network is that mathematical manipulations can be done with ease due to the matrix format. One disadvantage however, is that for a large sparse network, lots of memory is required to store the network as the relationship between a node and every other node is stored for all nodes [7].

A method of storage that gets around this issue is the edge list, where a list of edges is stored and nothing else. However, this is much harder to use when running an algorithm, while it also misses nodes that have no edges. The ideal choice of storage of a network largely depends on what the network will be used for.

A network can be split into a set of N communities $\mathbf{C} = (C_1, C_2, ..., C_N)$, where nodes V are partitioned into the communities [8]. These communities are expressed as different colours on the nodes in figure 1. Each community C_i has strong connections between the set of nodes within that community, with weaker connections to the rest of the network.

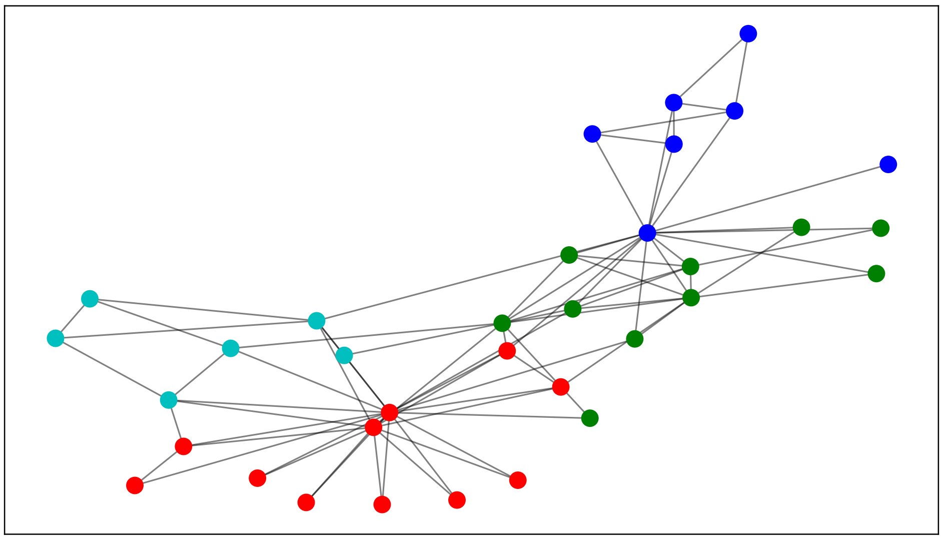

Figure 1: "Zachary's Karate Club" Network. This is a popular example of a network with a known community structure, after the club split in two following a disagreement [6]. Edges represent friendships, nodes represent members. Image produced by the author in python. The colours represent the community partition found using the Louvain algorithm.

The degree of a node k is defined as the number of edges attached to the node. Equation (2) represents the degree of simple networks.

$$k_i = \sum_j A_{ij}. \tag{2}$$

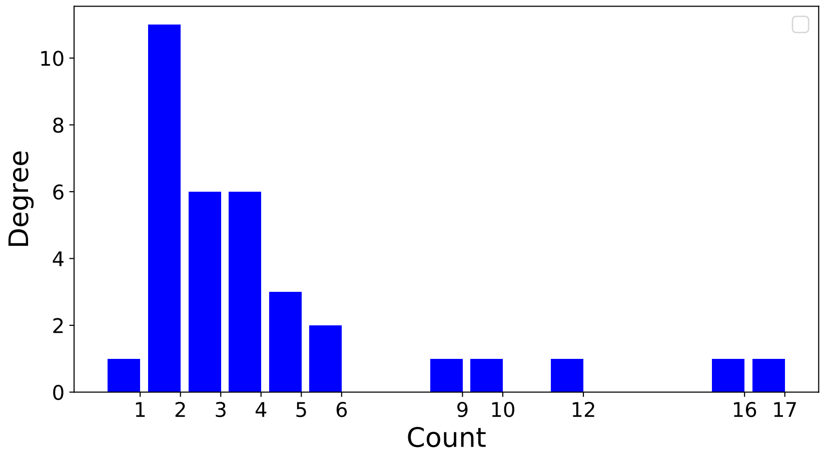

Figure 2: "Zachary's Karate Club" degree distribution [6]. The figure was created by the author using the matplotlib and networkx packages in python.

The configuration model creates a random graph however, the degree distribution is specified beforehand. An example of a degree distribution is shown in figure 2. At first, k_i

stubs are assigned to each node V_i, which the set of k_i reflecting the degree distribution. Following this, two stubs are selected at random and joined with an edge. These steps are then repeated until all stubs are joined.

A random walk across a network consists of a series of random steps. The walk first begins on a node V_i, where $k_i \geq 1$. One of the nodes that shares an edge with node V_i is picked randomly and the walk progresses to this node. This is repeated for L iterations at which point the walk ends.

3 Community Detection

Community detection is a particularly active area of network science that allows light to be shed on the structure of a large scale network. If communities are able to be identified within the structure of a network this information can be extremely useful [9] [10]. For example, it can be used by online retailers to identify users with increasingly similar shopping habits, and show those users targeted advertisements. Or it can be used to identify the probability of a given user voting for a political party in an upcoming election, based on who they associate with and how that group usually votes. This application has been of particular interest since the Brexit referendum and the US Election in 2016 [11].

There are several different types of community; Disjoint communities are non-overlapping, meaning that a node V_i can belong to only one group. Overlapping communities are similar to disjoint communities however, a node can belong to more than one community. Hierarchical communities show a structure that represents levels of grouping for example the UK government network would likely have this structure with the Prime Minister at the top. Finally, local communities have a local structure with no global structure [12].

In this essay, the algorithms looked at detecting disjoint communities. One kind of error on the detection could be that the wrong type of community was detected and this will be discussed in section 3.6. The networks used are also static, meaning that they are frozen in time, as opposed to dynamic networks that aren't [13].

3.1 Modularity

A community should have a large number of edges between nodes in the community, with a small number of edges from nodes in the community to nodes outside it. This is the idea behind modularity, defined in Eq (3).

$$Q = \frac{1}{2m}\sum_i\sum_j [A_{ij} - \frac{k_i k_j}{2m}]\delta_{c_i c_j}. \tag{3}$$

In Eq (3), c_i and c_j represent the communities in which nodes V_i and V_j are assigned to, k_i and k_j represent the degree of nodes i and j while m represents the total number of edges within the network.

The aim of modularity is to compare the number of edges inside a community, with the expected number of edges if we built a network with the same distribution using the configuration model.

If we consider two nodes v and w which have degrees k_v and k_w respectively. The total number of stubs in the network is equal to L and can be calculated by summing the degree of all the nodes, as seen in Eq (4).

$$L = \sum_i k_i, \tag{4}$$

As each edge has two nodes on either side, the total number of stubs L is equal to twice the number of edges m. Consequently, Eq (5) shows the relationship between the sum of the stubs and m.

$$L = 2m = \sum_i k_i. \tag{5}$$

If each of the k_v stubs on node v is considered, an indicator I_h is assigned for each stub h. If one of these stubs has an edge attached to it that joins to a stub on node w, $I_h = 1$, else $I_h = 0$. As per the configuration model, the stubs have an equal probability of connecting to the remaining $2m - 1$ stubs.

The stub on node v can connect to any of the k_w stubs on node w, hence Eq (6) shows the probability that $I_h = 1$.

$$P(I_h = 1) = \frac{k_w}{2m - 1}, \tag{6}$$

As there are k_v stubs on node v, the number of edges between v and w is J_{vw}, as seen in Eq (7). Taking the expectation of this value shows the expected number of edges in Eq (8)

$$J_{vw} = \sum_{h=1}^{k_v} I_h, \tag{7}$$

$$E[J_{vw}] = E[\sum_{h=1}^{k_v} I_h] = \sum_{h=1}^{k_v} E[I_h] = \frac{k_v k_w}{2m - 1},$$

$$E[J_{vw}] = \frac{k_v k_w}{2m - 1} \quad \text{as } 2m \gg 1. \tag{8}$$

If the number of expected edges between two nodes in a community is taken away from the number of actual edges, then summed across all nodes in each community, a measure of how strong the structure of communities is found. To look just at nodes within a community, $\delta_{c_i c_j}$ is multiplied by the subtraction. If the nodes V_i and V_j are in the same community, $\delta_{c_i c_j} = 1$, else $\delta_{c_i c_j} = 0$. Finally, a factor of $\frac{1}{2m}$ is multiplied by the modularity in order to normalise it, giving Eq (3) [14].

In order to identify communities within a network, one should, therefore, split the network into communities such that the modularity is the highest. However optimizing modularity has been shown to be NP-Complete [15], so algorithms that approximate the highest value must be used instead.

Modularity is a useful tool in identifying community structure however, it isn't perfect. By comparing the actual edges with the edges expected in the configuration model, it is assumed that each node can attach to any other node. In real life networks, this isn't always the case, as the likely horizon of a node often only includes a small part of the network [18].

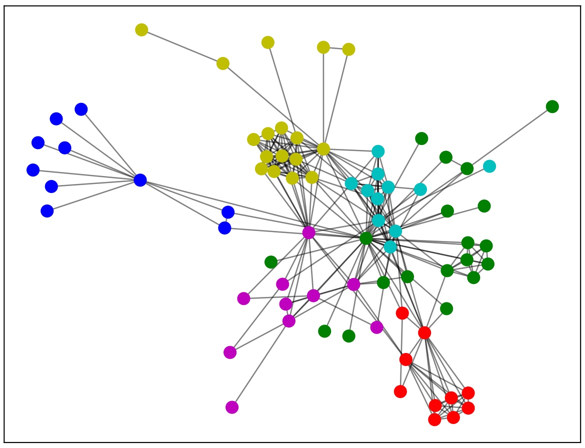
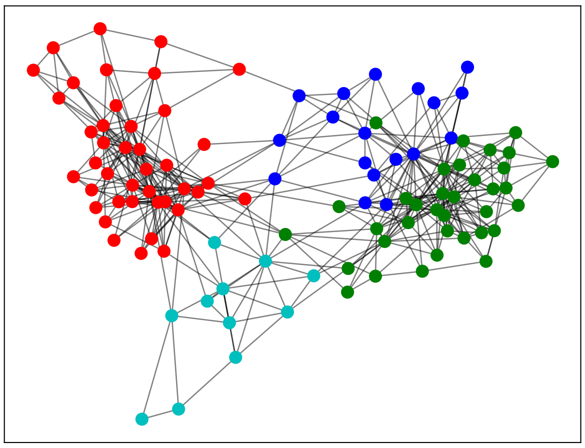

Figure 3: On the left is a network representing the co-appearance of characters in Les Miserables [16], divided into communities with $Q = 0.55$. On the right is a network representing books about US politics, where there is an edge present if a large number of people bought both books [17]. This network is also divided into communities with $Q = 0.53$. Each colour represents a community. Both graphs were visualized by the author using the community python package to implement the Louvain method in order to find the community structure and the networkx package in python to draw the graph.

The assumption will also mean that the expected number of edges between two groups of nodes will decrease as the size of the network increases. This is because these two groups can be treated as nodes with degrees k_v and k_w, as the size of the network increases, the number of edges m increases and consequently, the expected number of edges between the two communities falls as per Eq (8). As a result for large networks, the expected number of edges could be smaller than one, meaning that even a single connection between the two groups leads to an increase in modularity. Consequently, optimizing modularity in a large network won't resolve small communities, regardless how well defined that community is. This problem is known as the 'Resolution Limit of Modularity' [19].

3.2 Spectral Methods of modularity optimisation

Initially, a network of n nodes and m edges is presented, with the aim to divide the nodes into two groups to maximise modularity Q, shown in Eq (3). Equation (9) redefines the Kronecker delta $\delta_{c_i c_j}$, with $s_i = 1$ if node V_i is in group 1, and $s_i = -1$ if V_i is in group 2. The same is true for s_j with V_j [14].

$$\delta_{c_i c_j} = \frac{1}{2}(s_i s_j + 1). \tag{9}$$

The terms inside the brackets in Eq (3) are also redefined as shown in Eq (10). The elements of the 'Modularity Matrix' $\mathbf{B}$ are formed from Eq (10).

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}. \tag{10}$$

It can be shown that **B** has the property that the sum of both its rows and columns is zero, as is seen in Eq (11).

$$\sum_j B_{ij} = \sum_j A_{ij} - \frac{k_i}{2m} \sum_j k_j = k_i - k_i \frac{k_i}{2m} = 0. \tag{11}$$

Consequently Q can be expressed as in Eq (12), with the one cancelling out due rows and columns summing to zero as shown in Eq (11).

$$Q = \frac{1}{4m} \sum_i \sum_j B_{ij}(s_i s_j + 1),$$

$$Q = \frac{1}{4m} \sum_i \sum_j B_{ij}(s_i s_j). \tag{12}$$

The aim of the method is to maximise Q by choosing the groups that represent s_i. As previously discussed, this task is NP-Complete, so an approximation is needed [15].

The s_i can only be discrete values of either 1 or -1 however, this condition is relaxed by allowing them to take any real value. The constraint shown in Eq (13) is applied, however.

$$\sum_i k_i s_i^2 = \sum_i k_i = 2m. \tag{13}$$

By using a Lagrange multiplier, this optimization problem can now be solved.

$$\frac{\partial}{\partial s_i}\left(\sum_{ij} B_{ij} s_i s_j - \lambda \sum_i k_i s_i^2\right) = 0,$$

If the derivatives are done and the equation is rearranged, the result is Eq (14).

$$\sum_j B_{ij} s_j = \lambda k_i s_i, \tag{14}$$

Written in matrix notation, Eq (14) becomes Eq (15).

$$\mathbf{Bs} = \lambda \mathbf{Ds}. \tag{15}$$

Here **s** is a vector with elements s_i and **D** is a diagonal matrix with elements $D_{ii} = k_i$, the node degrees. The eigenvector equation shown in Eq (15) has an eigenvalue λ and an eigenvector **s**. If Eq (14) is multiplied by s_i, then summed over all nodes in community i, the result is an expression for modularity Q as shown below:

$$Q = \frac{1}{4m} \sum_{ij} B_{ij} s_i s_j = \frac{\lambda}{4m} \sum_i k_i s_i^2,$$

Using the constraint in Eq (13), this result becomes Eq (16).

$$Q = \frac{\lambda}{4m} 2m = \frac{\lambda}{2}. \tag{16}$$

From Eq (16) it is clear that modularity $Q \propto \lambda$. Consequently, the highest eigenvalue λ should be picked in order to maximise Q.

As all rows and columns of the modularity matrix $\mathbf{B}$ sum to zero, the eigenvector $\mathbf{s} = (1, 1, 1, ..., 1)$ with an eigenvalue $\lambda = 1$ is always a solution. This solution is saying that all of the nodes are in the first community and none are in the second. Consequently, if it is found that $\lambda = 0$ is the highest eigenvalue, then the highest modularity is achieved by not dividing the communities at all, a case known as an indivisible network.

Logically if there are any positive eigenvalues, then a solution exists that has a higher modularity than the undivided network. This property is assumed in the next steps of this method.

Using the definition of the modularity matrix $\mathbf{B}$, Eq (14) can be re-written as Eq (17).

$$\sum_j A_{ij}s_j = k_i(\lambda s_i + \frac{1}{2m}\sum_j k_j s_j), \tag{17}$$

In matrix notation Eq (17) becomes Eq (18), where $\mathbf{1}$ is a vector with the same dimensions as $\mathbf{s}$, with $1_i = 1$ for all i.

$$\mathbf{As} = \mathbf{D}(\lambda\mathbf{s} + \frac{\mathbf{k}^{\intercal}\mathbf{s}}{2m}\mathbf{1}), \tag{18}$$

Note that due to the previously outlined properties of the matrices:

$$\mathbf{A1} = \mathbf{D1} = \mathbf{k},$$

As a result, if Eq (18) is multiplied by $\mathbf{1}^{\intercal}$, the result is that:

$$0 = \lambda\mathbf{k}^{\intercal}\mathbf{s},$$

As having at least one nontrivial solution with high modularity is assumed, Eq (19) must hold as $\lambda \neq 0$. This means that Eq (18) then simplifies to Eq (20).

$$0 = \mathbf{k}^{\intercal}\mathbf{s}, \tag{19}$$

$$\mathbf{As} = \lambda\mathbf{Ds}. \tag{20}$$

In order to maximise Q, the largest eigenvalue λ that is allowed should be used. The vector $\mathbf{1}$ can be used as an eigenvector. By the 'Perron Frobenius' theorem [20], as all of its elements are positive, it must have the most positive eigenvalue. This choice is forbidden, however as it doesn't satisfy Eq (19). So the next most positive eigenvalue is chosen. This is also the same solution as the highest eigenvalue in Eq (15).

Now that a solution is obtained for our relaxed problem, it needs to be translated to the unrelaxed discrete problem. One approach is to round the s_i to the nearest discrete value as $s_i = \pm 1$. Therefore for positive values $s_i = 1$ and for negative values $s_i = -1$.

So in summary, the algorithm to split a network into two communities is to calculate the eigenvector $\mathbf{s}$ in Eq (20) that corresponds to the second-highest eigenvalue, then divide the groups accordingly based on the signs of the eigenvector. This gives an approximate solution in how best to divide the network.

The final step is to keep repeating the steps described above, with the difference that the network that is looked at is only comprised of the individual communities that have been previously detected. This enables the original network to be split up into more than two communities, but a check must be performed to make sure that this action doesn't cause modularity to decrease compared to if nothing was changed

3.3 Louvain Method

The Louvain method a stochastic process that aims to optimise modularity Q as described in Eq (3). It can work on weighted or unweighted networks, where an unweighted network the weights of all edges can be taken to be equal to one. During this essay, I will assume that an unweighted network is being dealt with. The algorithm is divided into two phases, which are then repeated [21].

Starting with a network of N nodes, a different community is assigned to each node. If a node V_i is considered, the algorithm calculates the ΔQ if V_i were to be placed in a community j, where j can be the community of any other node. Following this, V_i is then put in the community where ΔQ is the most positive. However, if $\Delta Q < 0$ for all possible changes, V_i stays in its original community.

This process is then repeated across all the nodes until no further positive ΔQ is possible. Once this has happened phase one of the algorithm is complete and a local maximum of Q is achieved.

Note that the order in which the nodes are looked at can have an effect on the final arrangement of nodes following the conclusion of phase 1, making the process stochastic. This order can also have effects on the time the computational time needed.

ΔQ can be calculated using Eq (21) [22], where $\sum_{in}$ is the sum of the weights of the edges within a community C, $\sum_{tot}$ is the sum the weights of the edges incident to nodes within community C. $k_{i,in}$ is the sum of the weights from V_i to nodes in C. m is the total sum of all the weights in the network and k_i is the sum of the weights from V_i. In this phase, as an unweighted network is being dealt with the weights are all set to one.

$$\Delta Q = [\frac{\sum_{in} + k_{i,in}}{2m} - (\frac{\sum_{tot} + k_i}{2m})^2] - [\frac{\sum_{in}}{2m} - (\frac{\sum_{tot}}{2m})^2 - (\frac{k_i}{2m})^2]. \tag{21}$$

Phase 2 builds a new network, with the nodes in this network being the communities of the first phase. The weights of the edges are proportional to how many edges there are between nodes in the two communities. Edges between nodes in the same community lead to self-loops within the new network. The first phase is then re-applied to this new network until a local maximum of modularity is found. Following this, the whole process is repeated until no further increases in modularity can be gained.

The number of passes is defined as the number of times creating a new network as outlined above creates a modularity increase.

If this algorithm is compared to the Spectral method, it can be seen that there are several advantages [21]:

1. The algorithm is easier to implement in code.

2. Due to the use of Eq (21), the algorithm is very fast.

3. The Resolution Limit of Modularity, as discussed in section 3.1 is partially overcome due to the multi-level nature of the algorithm.

4. The intermediate stages of the algorithm can give information about community micro-structure within a network.

3.4 InfoMap

Another approach is to use concepts from Information theory. Consider a random walk on a network. The order of the nodes visited will give information about the structure of the network. For example, two nodes that are consecutively visited must be joined with an edge. Information about community structure will also be contained, as communities will ideally have lots of edges within them, with few edges to other communities. Therefore if the walk is inside a community, it tends to stay there as there are few edges to escape to other communities. It is assumed the walk is long enough to cross all the nodes.

Rosvall and Bergstrom [23], provide a way to quantify this by turning the walk into a binary bit-string, a sequence of ones and zeros that describes the walk. If the network is divided into communities (groups), each community is given an entrance and an exit label. Every time the walk enters a community, the entry label is recorded. Once the walk exits the community the exit label is recorded. Within a community, the walk is defined by giving each node a label and recording that label each time the node is visited. This is shown in figure 4. Once the walk is finished, the sequence of node, entry and exit labels forms the bit-string.

The InfoMap method works on the idea that the best division of the network will have the shortest bit-string. The length of the bit-string, depends on the length of the labels themselves, so all the possible ways of labelling the nodes and communities are considered and the one chosen is the one that gives the shortest bit string for a particular division. This is related to the problem in information theory of compressing information, e.g. a file on a computer [23].

The node labels within communities can be the same as the node labels in other communities, as the current community is always known due to the entry and exit labels. The exit labels can also be the same for all communities, as they only have to be different from the node labels in that particular community. The node labels also don't have to be the same length, as it would make sense to shorten the labels for nodes that are visited the most often in a community.

For a good division, there would be few edges between groups, meaning that the walk wouldn't change communities often. For this reason, in the bit string, fewer entry and exit labels would be recorded resulting in a shorter bit-string. If the groups are also small, the length of the labels within the groups can be kept short, as for b bits there are 2^b distinct labels. On average for a group with N nodes, $b \approx \log_{10} N$ bits would be needed for distinct labelling. It's possible to do slightly better than this by using shorter labels for the most visited nodes. Both these facts mean that to get the shortest bit string, a balance will need to be found between having a few large groups and lots of small groups [23].

The standard algorithm is:

1. Generate a random walk that visits all parts of the network.

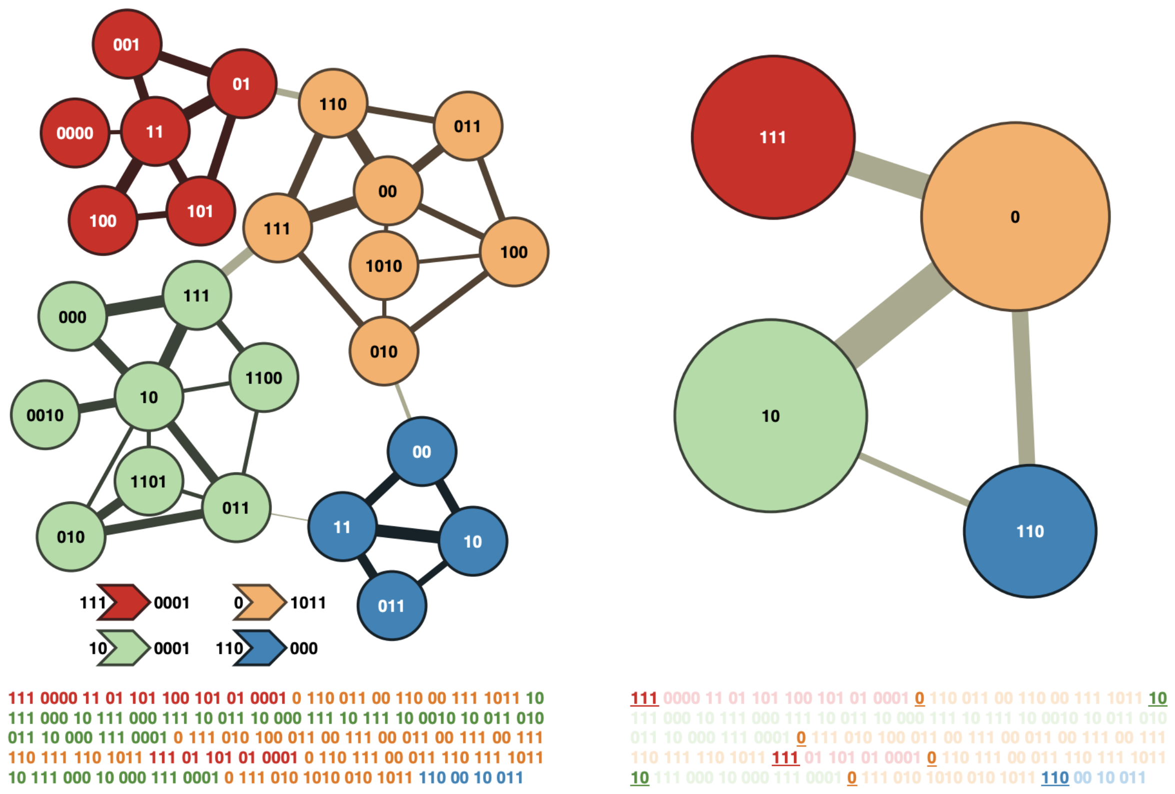

Figure 4: On the left is a two-level description of a network. Major communities are represented by colours, as well as their entry and exit labels, shown near the bottom. Nodes within communities also have a label. On the right, only the communities are represented, each with a label. The bottom shows an example bit-string for a random walk on the network, where the walk starts on node 0000. From Rosvall and Bergstrom [23].

2. For each possible division of the network, find the set of group and node labels that gives the shortest representation of the walk as a bit-string.

3. Find the division that has the shortest bit-string overall. This is the best community structure.

This however, is extremely slow. To speed the algorithm up, a result of information theory is used. Shannon's source coding theorem [24], states that for the shortest possible bit string the average number of bits per step of a walk L is equal to the entropy of that walk. This is governed by Eq (22), the map equation.

$$L = qH(Q) + \sum_s p_s H(P_s). \tag{22}$$

The quantity q is the fraction of time the walk spends going between communities, p_s is the fraction of time it spends within a community s. $H(X)$ describes the entropy of a sequence of nodes X, shown in Eq (23). The fraction of times a node V_i is in the sequence is represented by X_i.

$$H(X) = -\sum_i X_i \log_2 X_i. \tag{23}$$

Hence $H(Q)$ is the entropy of the sequence of groups from the walk and $H(P_s)$ is the entropy of the nodes in a group s.

These results mean that the average length of the walk L can be calculated without the labels being assigned to the nodes or communities and without the random walk happening at all. The quantities $p_s, H(Q), q$ and $H(P_s)$ can be calculated by knowing the structure of the local network and the fraction of time the walk spends at each node, which is proportional to the degree of that node k [25]. An algorithm similar to the one used in the Louvain method from section 3.3 is used to minimise L [23].

This method is quite different from the modularity based methods previously explored however, both types ultimately turn into an optimisation problem.

3.5 Other methods

Another category of community detection method are statistical interference based methods. Some of these methods use the Stochastic Block Model (SBM). In general, the model also groups nodes into non-overlapping communities [26]. These methods, however need the number of communities to be known in advance. This assumption is unsuitable for a real-world network, as this a parameter that needs to be found. For this reason, this category of method will not be looked at any further.

Dynamic community detection shows the evolution of communities over time. This is useful in disciplines in which data is frequently being updated such as social networks. One approach to doing this is to first run a static algorithm on the first version of the network. Following this, the community structure is re-identified using the same static algorithm for each update of the network. The static algorithm is usually modified in order to introduce a penalty for the global structure of the communities deviating significantly in a short time window to increase stability. An example is the TILES algorithm for dynamic social networks [27].

Furthermore as discussed above, the algorithms investigated so far only search for disjoint communities. To find hierarchical clusters, one method is to use similarity measures [28]. There are various methods to measure similarity, which is both a strength and weakness of this method. The similarity between all pairs of nodes is calculated, after which the nodes with the highest similarity are joined together. Following this, the group's similarity is calculated and then they are joined together and so on. There are various different measures of group similarity, with one of the most widely used being single linkage clustering, with the similarity between two different groups defined as the similarity of the most similar nodes in those two groups [29]. Eventually, all nodes will be combined into one group, which will show the hierarchy of the network [2].

Other community detection algorithms exist for networks with overlapping communities, such as the non-negative matrix factoring approach [30].

3.6 Errors and Uncertainties

Very little is discussed with regards to errors and uncertainties on community detection within networks, as it is a field in its infancy. The methods outlined in this essay give one definite answer for the community structure of a network, but don't give any uncertainties on that community structure. There are several possible methods for estimating the uncertainty on the community structure, however.

For stochastic methods such as the Louvain method outlined in section 3.3, different results can be obtained by running the algorithm multiple times on the same network. Therefore one way to test the stability of the community structure is to run the algorithm a set number of iterations, then record the proportion of nodes that are in different communities for each iteration. This will give an idea as to how stable the community structure is.

Another method of testing the reliability of methods is to find a network with a known community structure, then compare that structure to the structure found when a community detection algorithm is run on the network [2]. An algorithm can then be optimized by trying to reproduce the existing community structure on the network. If this is done, it means that the algorithm is now only valid for that particular network though, as the community structure in that network may not be representative of the community structure in general networks.

Another potential source of uncertainty is that as outlined in section 3.1, modularity Q is not a perfect indicator of community structure, so even if modularity is at its maximum value it doesn't necessarily mean that the community structure is correct [31]. The nature of certain networks, e.g. social networks will mean that the correct community structure is somewhat open to interpretation and therefore can't be mathematically quantified.

Consequently, the error on that community structure also can't be quantified. In some cases, the best course of action is to compare the community structure produced by a variety of methods and come to a consensus view.

4 Applications of Community Detection

4.1 Social Network Analysis

The field of social network analysis is continually growing both in the amount of data and its applications [32]. Over the last decade, there has been a steep rise in the number of users of online social networks so consequently far more data is available. Ego networks are often used here, which focus on a central node [33]. For example, a LinkedIn Ego network could focus on all my connections and the connections between those connections. In some networks, the central node is removed to give the graph a clearer format.

Community detection can be used here for a variety of purposes [34], such as the detection of terrorist groups. If a suspected terrorist is identified, community detection algorithms can be deployed across the suspect's social media in order to work out if and who that person is working with [35]. This application is very useful however, it can be misused if the authority doesn't verify the results. As is discussed in section 3.6, these algorithms give only one answer which can change depending on a variety of factors [8].

4.2 Link Prediction

Another interesting application of social network analysis is the prediction of missing edges (links) within networks. Community detection is an important step as it shows the general structure of the network. Once this structure is identified, the probability of an edge existing between two nodes can be calculated. There has been work that suggests that link detection is significantly improved when the underlying community structure is known [36].

4.3 Smart Advertising

Targeted adverts can be done if a company knows who they are talking to. Using community detection algorithms, a user's community can be identified. Following this, the user can be shown specific advertisements that have previously resonated with the users in the same community. The online customer experience can be improved as the customers are shown products and services that are likely to be relevant to them [8].

5 Testing of methods

5.1 Existing Community Structure Comparison

A commonly used example is that of "Zachary's Karate Club" [6], shown in figure 5. This network was constructed from a university karate club, just before the club split in two, on factional grounds. There were two different ideologies in the club, which eventually led to disagreements at club meetings. The nodes represent members of the club, while the edges represent friendships that were observed between two members of the club, outside of usual club activities. In the original paper, the club that each of the club members

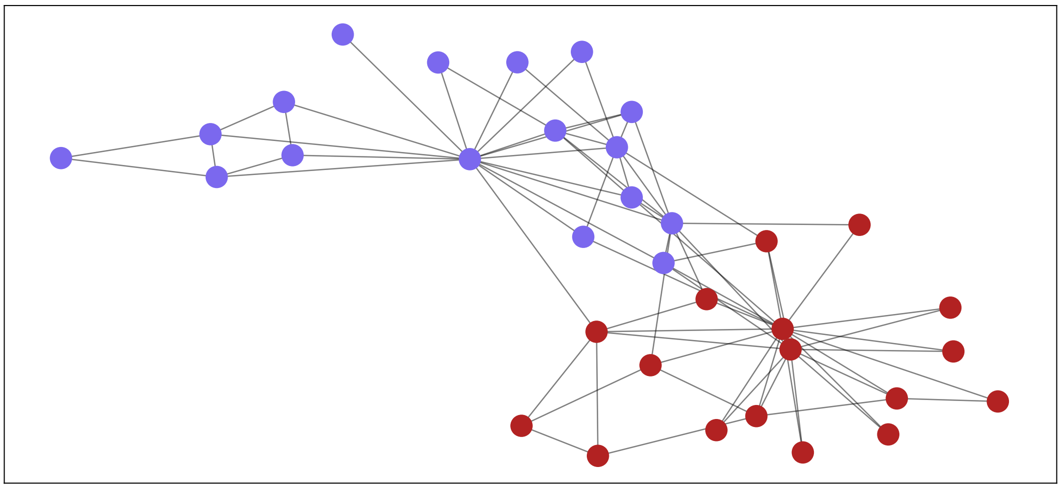

Figure 5: "Zachary's Karate Club" network, where the colours show the actual community division within the club [6]. The figure was created by the author using the matplotlib and networkx packages in python.

joined after the split is recorded, so if the algorithms are run on the network the broad network structure results can be compared to the recorded split [6]. Note that there may be more sub-communities within the network so for the purposes of this comparison these communities will be grouped to find two broad communities.

The Spectral algorithm from section 3.2, the Louvain method from section 3.3 and the InfoMap method from section 3.4, were applied to the Karate network. Once the subgroups had been identified and combined, the proportion of nodes in the correct community was identified. As the Louvain method is stochastic, it was run five times and the results were averaged. The results are summarised in the table below:

Algorithm	Proportion Correct
Spectral	94.1%
Louvain	95.9%
InfoMap	94.1%

The results show that for this particular graph, the Louvain method was slightly better at predicting the inherent community structure however, the difference is marginal. This result isn't valid for general graphs though, as the conclusion reached here is only applicable to the Karate Club network.

5.2 Speed and Maximum Modularity

The aim of the two methods is to maximise the modularity of the community structure on a network, so it will make sense to compare how well they do this. Another important consideration is speed. If the algorithm gains a 1% better modularity score on average but it is 10x as slow as another algorithm, it may not necessarily be more suitable.

In order to compare the algorithms, a variety of networks was tested with varying numbers of nodes and edges. This was done on a 2017 Apple MacBook Pro using the community python package implementation of the Louvain and Spectral algorithms. The InfoMap algorithm was implemented using the python package InfoMap. The tests were repeated 5 times and then an average value was taken except for cases where the test took longer than 6 hours, in which case the test was abandoned.

The first network that the algorithms were applied to was Zachary's Karate Network [6]. The next was the US political books network [17] that's shown in figure 3. Following this was the network from Les Miserables [16], also shown in figure 3. Following this, a network of Facebook pages for politicians in the US is tested [37]. Finally, a Facebook ego network was tested [38].

Network	Nodes/Edges	Spectral Method: Q/t	Louvain Method: Q/t
Karate Club	34/78	0.42/2.92s	0.42/0.03s
US Political Books	105/441	0.53/31.4s	0.53/0.06s
Les Miserables	77/254	0.55/18.5s	0.56/0.06s
Politician Facebook	7057/89455	N/A/+6h	0.72/11.5s
Ego Facebook	4039/88234	N/A/+6h	0.83/10.2s

From the results above it is clear that the Louvain method is significantly faster than the spectral method, both for relatively large and small networks. It is difficult to conclude which algorithm is better for optimising modularity, however. The results show that for the networks where data was able to be collected for both methods, the modularity is approximately equal. On larger networks, it is clear to see that the Louvain method is more suitable, due to its superior speed.

In our data, the modularity tends to be bigger for the larger networks. Further investigations would check if this is a trend or specific to the networks that are seen in this essay. Note that the InfoMap method does not aim to maximise the modularity of the network, so it would make no sense to compare its modularity.

Next, the time taken to determine the community structure using the InfoMap method from section 3.4 was recorded for the same networks as before. The results are summarised below:

Network	Nodes/Edges	t
Karate Club	34/78	0.001s
US Political Books	105/441	0.004s
Les Miserables	77/254	0.002s
Politician Facebook	7057/89455	0.406s
Ego Facebook	4039/88234	0.256s

Looking at the time taken, it is clear that the InfoMap method is significantly faster than both of the modularity based methods. However, this speed difference could be partially explained due to the efficiency of the implementations in python. It can be concluded that the fastest method tested across every example network was the InfoMap method.

Initially, these tests were all supposed to be done on ego networks generated from the author's LinkedIn and Facebook profiles. This networked data was previously freely downloadable, however since 2017 this is no longer the case. The data can now only be obtained via a web scraper, which violates both the site's terms of service. Ethical concerns have meant that in the recent past other people's data is being made harder to obtain. This is to stop this data being used in ways the original producer would not intend.

6 Conclusion

The core aim of the project was to explore the methods and applications of community detection in networks has been accomplished. Three algorithms for community detection have been investigated, Spectral Optimisation, the Louvain method, and the InfoMap method.

Finally, the methods were tested on real data-sets in python in order to compare them. On the Karate Club Network, it has been found that the Louvain method was the most effective in finding the established community structure. The InfoMap method was found to be the fastest method to divide a graph into communities on every network tested.

Further investigations would focus on exploring more applications of community detection and networks in general. For example the uses of community detection in modelling the spread of epidemics on networks [39] and exploring different types of community detection algorithms, as touched on in section 3.5. The relationship between the size of the network and the maximum modularity could also be investigated in further detail.

References

[1] S. Fortunato and D. Hric, "Community detection in networks: A user guide," *Physics reports*, vol. 659, pp. 1–44, 2016.

[2] M. Newman, *Networks.* Oxford university press, 2018.

[3] H. Zenil, N. A. Kiani, and J. Tegnér, "Quantifying loss of information in network-based dimensionality reduction techniques," *Journal of Complex Networks*, vol. 4, no. 3, pp. 342–362, 2016.

[4] T. Evans, "Complex networks," *Contemporary Physics*, vol. 45, no. 6, pp. 455–474, 2004.

[5] A.-L. Barabási *et al.*, *Network science.* Cambridge university press, 2016.

[6] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of anthropological research*, vol. 33, no. 4, pp. 452–473, 1977.

[7] G. Caldarelli and M. Catanzaro, *Networks: A very short introduction.* Oxford University Press, 2012, vol. 335.

[8] A. Karataş and S. Şahin, "Application areas of community detection: A review," 12 2018, pp. 65–70.

[9] J. O. Garcia, A. Ashourvan, S. Muldoon, J. M. Vettel, and D. S. Bassett, "Applications of community detection techniques to brain graphs: Algorithmic considerations and implications for neural function," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 846–867, 2018.

[10] F. Guerrero-Solé, "Community detection in political discussions on twitter: An application of the retweet overlap network method to the catalan process toward independence," *Social science computer review*, vol. 35, no. 2, pp. 244–261, 2017.

[11] B. Edmonds, "Co-developing beliefs and social influence networks—towards understanding socio-cognitive processes like brexit," *Quality & Quantity*, pp. 1–25, 2019.

[12] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[13] D. R. Farine, "When to choose dynamic vs. static social network analysis," *Journal of Animal Ecology*, vol. 87, no. 1, pp. 128–138, 2018. [Online]. Available: https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/1365-2656.12764

[14] M. E. Newman, "Spectral methods for community detection and graph partitioning," *Physical Review E*, vol. 88, no. 4, p. 042822, 2013.

[15] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner, *On modularity-np-completeness and beyond.* Univ., Fak. für Informatik, Bibliothek, 2006.

[16] D. E. Knuth, *The Stanford GraphBase: a platform for combinatorial computing.* AcM Press New York, 1993.

[17] V. Krebs, "A network of books about recent us politics sold by the online bookseller amazon. com," *Unpublished http://www. orgnet. com*, 2008.

[18] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, "Mixing local and global information for community detection in large networks," *Journal of Computer and System Sciences*, vol. 80, no. 1, pp. 72–87, 2014.

[19] S. Fortunato and M. Barthelemy, "Resolution limit in community detection," *Proceedings of the national academy of sciences*, vol. 104, no. 1, pp. 36–41, 2007.

[20] K.-C. Chang, K. Pearson, and T. Zhang, "Perron-frobenius theorem for nonnegative tensors," *Communications in Mathematical Sciences*, vol. 6, no. 2, pp. 507–520, 2008.

[21] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, "Generalized louvain method for community detection in large networks," in *2011 11th International Conference on Intelligent Systems Design and Applications.* IEEE, 2011, pp. 88–93.

[22] 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*, vol. 2008, no. 10, p. P10008, 2008.

[23] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, 2008.

[24] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[25] J. D. Noh and H. Rieger, "Random walks on complex networks," *Physical review letters*, vol. 92, no. 11, p. 118701, 2004.

[26] E. Abbe and C. Sandon, "Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery," in *2015 IEEE 56th Annual Symposium on Foundations of Computer Science.* IEEE, 2015, pp. 670–688.

[27] G. Rossetti, L. Pappalardo, D. Pedreschi, and F. Giannotti, "Tiles: an online algorithm for community discovery in dynamic social networks," *Machine Learning*, vol. 106, no. 8, pp. 1213–1241, 2017.

[28] J. Irani, N. Pise, and M. Phatak, "Clustering techniques and the similarity measures used in clustering: a survey," *International Journal of Computer Applications*, vol. 134, no. 7, pp. 9–14, 2016.

[29] L. McInnes, J. Healy, and S. Astels, "hdbscan: Hierarchical density based clustering," *Journal of Open Source Software*, vol. 2, no. 11, p. 205, 2017.

[30] J. Yang and J. Leskovec, "Overlapping community detection at scale: a nonnegative matrix factorization approach," in *Proceedings of the sixth ACM international conference on Web search and data mining*, 2013, pp. 587–596.

[31] J. Xiang, T. Hu, Y. Zhang, K. Hu, J.-M. Li, X.-K. Xu, C.-C. Liu, and S. Chen, "Local modularity for community detection in complex networks," *Physica A: Statistical Mechanics and its Applications*, vol. 443, pp. 451–459, 2016.

[32] L. Freeman, "The development of social network analysis," *A Study in the Sociology of Science*, vol. 1, p. 687, 2004.

[33] R. A. Hanneman and M. Riddle, "Introduction to social network methods," 2005.

[34] S. Wasserman, K. Faust *et al.*, *Social network analysis: Methods and applications.* Cambridge university press, 1994, vol. 8.

[35] I. Gialampoukidis, G. Kalpakis, T. Tsikrika, S. Papadopoulos, S. Vrochidis, and I. Kompatsiaris, "Detection of terrorism-related twitter communities using centrality scores," in *Proceedings of the 2nd International Workshop on Multimedia Forensics and Security*, 2017, pp. 21–25.

[36] J. C. Valverde-Rebaza and A. de Andrade Lopes, "Link prediction in complex networks based on cluster information," in *Brazilian Symposium on Artificial Intelligence.* Springer, 2012, pp. 92–101.

[37] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," 2019.

[38] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in *Advances in neural information processing systems*, 2012, pp. 539–547.

[39] M. Salathé and J. H. Jones, "Dynamics and control of diseases in networks with community structure," *PLoS computational biology*, vol. 6, no. 4, 2010.