An in-Depth Look at Modularity Methods for Community Detection

Networks

Candidate Number: 1061996

1 Introduction

Networks are used in a variety of fields as they give us insight into complex systems. One of the most useful insights we can gain from networks is which collections of nodes form communities: densely connected groups of nodes with more sparse connections to the rest of the network. These are useful to find as they help us explain network phenomena [8, 9]. Communities can be considered fairly independent components of networks and therefore nodes in the same community often play a similar role, for example groups of friends in social networks may be interested in similar things [5, 9].

In disciplines where information is frequently represented as graphs, such as biology or computer science, this problem is of great importance. As such, the process known as community detection has garnered a lot of attention in recent years. Despite the problem seeming easy visually, to define it in mathematical terms and implement methods to detect communities in graphs, which can contain billions of nodes, has proved a challenge [5].

Nonetheless, developments have been made over the last 20 or so years with one of the most popular methods of community detection being modularity methods [5, 9]. These methods centre around the optimisation of an objective function, called the modularity, over partitions of the data into communities. Although modularity maximisation is popular, there have been papers suggesting the output of these methods should be interpreted cautiously when applied to real-world networks [6, 7].

In this essay, we plan to investigate modularity methods starting from first principles. We will discuss how they are used to find communities in networks, how modularity methods have inherent issues as a result of the way modularity is defined, and how methods have been developed to deal with these issues. Ultimately, we will build up to how modularity methods are being used in recently released papers and we will run some numerical simulations to back up the theory and compare different methods.

2 Community Detection and Modularity

Now we have an idea of what community detection and modularity are we are going to define them more rigorously. This section will serve as a mathematical introduction to what modularity is and how it can be used for community detection. We will mainly be considering unweighted, undirected networks in this essay but note that much of this theory can be extended to more complex cases fairly easily.

It is also important to note that to identify communities (also known as modules) accurately we will need the network to be sparse. That is, we will need the total number of edges in the graph to be of the order of the number of nodes in the graph. If there are too many edges then the distribution of edges will be too homogeneous to identify meaningful communities and we will end up finding "communities" which are in fact unrelated groups of nodes [5].

2.1 Motivation, Definitions and Notation

Let us define a graph $\mathcal{G}$ of order $|\mathcal{G}| = n$ and denote the nodes by v_i for all $i \in \{1, \ldots, n\}$. Each graph can be represented as an $n \times n$ adjacency matrix A where $A_{ij} = 1$ if node v_i is adjacent to node v_j and $A_{ij} = 0$ otherwise. We define the total number of edges in the graph to be m and vertex v_i to have degree k_i for all $i \in \{1, \ldots, n\}$. We will call an assignment of each node to a specific group or community a partition.

As we mentioned in the introduction, giving a quantitative definition of community can be difficult hence the definition is different throughout the literature. However, the intuition is normally the same: we want more edges linking nodes within the community than edges linking from nodes within the community to nodes outside of the community [5].

Modularity was developed as a metric to quantify the quality of partitions found by community detection algorithms. It worked on a basic principle: random graphs would not be expected to exhibit any community structure so if our graph has a greater density of edges in a subgraph than we would expect a random graph to have in an equivalent subgraph we would expect there to be some community structure [11]. To define this random graph and expected density of edges we need to define a null model.

We want the null model to keep some of the structural properties without keeping the community structure. In particular, keeping a broad degree distribution is important in the structure and function of real networks so we want to keep the degree distribution the same. As such, a simple model is used in which the expected degree distribution (after averaging over all possible configurations of the model) matches the actual degree distribution of the graph. Hence, for an instance of our model, we have n vertices each with the same number of stubs emanating from them as their degree is in the actual graph. That is to say vertex i has k_i stubs coming from it. In order for there to be an edge between node i and j a stub from each node must join together. We assume that a stub is equally likely to connect to any other stub and since we have $2m$ stubs the probability that any stub will link to node j is given by $k_j/2m$. We can then extend this to say the probability that node i connects to node j is given by $k_ik_j/2m$, by additivity [5].

The modularity then gives the difference between the density of edges inside a subgraph and the expected density of edges an equivalent subgraph would have in the null model. More formally, if we define the expected number of edges between vertices i and j in the null model by P_{ij} and the cluster node i belongs to by g_i with $g_i \in \{1, \dots, n_c\}$ since there are n_c clusters of nodes, the modularity Q is given by

$$Q := \frac{1}{2m}\sum_{ij}(A_{ij} - P_{ij})\delta(g_i, g_j) \tag{1}$$

$$= \frac{1}{2m}\sum_{ij}(A_{ij} - \frac{k_ik_j}{2m})\delta(g_i, g_j) \tag{2}$$

where the sum runs over all pairs of vertices. The δ-function is 1 if v_i and v_j are in the same community ($g_i = g_j$) and is 0 otherwise.

Given that the delta function is zero for all pairs of nodes not in the same cluster if we let C_i be the ith cluster of nodes for $i \in \{1, \dots, n_c\}$ the definition can be written

$$Q = \frac{1}{2m}\sum_{c=1}^{n_c}\left[\sum_{v_i, v_j \in C_c}\left(A_{ij} - \frac{k_ik_j}{2m}\right)\right] \tag{3}$$

$$= \sum_{c=1}^{n_c}\left[\frac{l_c}{m} - \left(\frac{d_c}{2m}\right)^2\right] \tag{4}$$

where l_c is the number of edges connecting nodes in cluster c and d_c is the sum of the degrees of all nodes in cluster c [5]. Now, looking at equation (4) we can see that the first term in the summand represents the proportion of edges between nodes in the cluster compared to whole graph and the second term gives the proportion of edges that we would expect from the null model. Hence, we see that there is clustering of nodes together if the number of internal edges of the subgraph is greater than the expected number. The more positive this is, the more well defined clustering there is. Therefore, by assumption, the larger the value of the modularity on a partition, the better the partition is at splitting the nodes into communities in this sense. This would suggest using the modularity as an objective function to maximise over all possible partitions [5].

We know from lectures that modularity ranges in $[-0.5, 1]$ and if we partition the network into one community we get a modularity of 0. When maximising modularity we therefore want to get a value as close to 1 as possible and we should never get a value less than 0.

Suppose we want to divide n nodes into n_c non-empty groups. This is a combinatorial problem with the answer given by the Stirling number of the second kind $S_n^{(n_c)}$. Suppose we now want to divide the nodes into less than or equal to n_c groups so groups can be empty. The number of distinct community divisions is given by the sum of the Stirling numbers. Although the solution is not known in closed form we can bound it from below using the first two Stirling numbers so $\sum_{n_c=1}^{n} S_n^{(n_c)} \geq S_n^{(1)} + S_n^{(2)} = 2^{n-1}$ for all $n > 1$. Therefore it is not feasible to maximise modularity for a graph with many vertices in this way as the time required to do so grows exponentially with the order of the network. Thus, we need a new method to maximise modularity while keeping computation time lower [10].

There have been many algorithms designed using a range of different principles to solve this problem. Greedy algorithms are used to maximise the modularity as they often produce partitions with relatively high modularity in an relatively short amount of time when compared to other methods [5]. That is not to say that other methods are not used, for example spectral methods or simulated annealing are both popular alternatives. We have already seen the Louvain algorithm in lectures which is a greedy algorithm in the sense that it minimises locally at each step. One of the advantages of this is that step-by-step the partition is optimised starting with a very fine structure of n communities and ending up with a coarse-grained description. This allows us to have a look at the partition after each step and possibly find hierarchical structure [10]. That is, we can sometimes see when more than one smaller community is contained in a larger community. For example, if the nodes represented people and the edges friendships a community might indicate people who are fans of football. Sub-communities of this might be fans of different football teams and this gives a hierarchical structure to the network.

Here, we mentioned how algorithms could be developed to deal with the problem that not all partitions could be checked individually. In the next subsection, we will see how the definition of modularity in itself will cause us problems when searching for the best partition of a network into communities and how we can deal with that.

2.2 Modelling Issues

Throughout the last subsection we assumed that a partition with high modularity was inherently a good partition. However, the notion we have of being a good partition may not coincide with the definition of a high modularity partition in the way we had hoped.

The first problem with modularity that we will investigate will be that there is often a resolution limit for modularity maximisation models. That is to say that communities smaller than a certain size, which depends on the total number of edges in the network m, may fail to be resolved. This is a result of them being combined with larger clusters of nodes as it can cause an increase in modularity [7].

We will then look at the problem caused by the modularity function exhibiting degeneracies. This means that the function often admits exponentially many partitions with high modularity and no clear global maximum, which is clearly a problem for our maximisation algorithms [7].

Another smaller problem which we will not discuss in detail is that in the asymptotic limit of an increasingly modular network the height of the modularity function converges to 1. Once we know this it is not as much of a problem but it is important to be aware that this is why we cannot compare the modularities of different sized networks [7]. The combination of these three problems casts doubt on the reliability of modularity maximisation in scientific contexts [5, 7].

Ultimately, we want to identify community structure in networks and modularity maximisation is a model developed to do that. As with any model, it has flaws but work has been done to address these. In the rest of the essay, we will go on to explain why each of these phenomena occur, look at newer models for mitigating the negative effects of them and then run numerical simulations to show how the community detection is improved on actual networks.

3 The Resolution Limit

3.1 Introduction to the Resolution Limit

The first problem with modularity methods that we will investigate will be the resolution limit. As mentioned earlier, clusters of nodes below a certain size will not be identified as a community by the modularity function. This size depends on the overall number of links in the network and we will investigate what this limit is, see examples of where distinct communities are merged and discuss why it happens in this subsection. We will mainly be following the paper Resolution Limit in Community Detection [6] by Fortunato and Barthélemy.

Recall our graph $\mathcal{G}$ and the definition of modularity, specifically equation (4) from the previous section. Some important intuition for this subsection is that the first term in the summand represents the fraction of links joining nodes in module C_c, whereas the second term represents the expected fraction of links in the equivalent module formed by the null model we defined earlier. If a subgraph $\mathcal{S}$ is such that the first term is larger than the second then there are many more links in the subgraph than we would have expected (as quantified by our null model), hence $\mathcal{S}$ is a module.

As such, using the summand from equation (4), we say that a subgraph $\mathcal{S}$ forms a module if

$$\frac{l_{\mathcal{S}}}{m} - \left(\frac{d_{\mathcal{S}}}{2m}\right)^2 > 0. \tag{5}$$

Now, if we express the number of edges joining nodes in $\mathcal{S}$ to the rest of the network $l_{\mathcal{S}}^{out}$ as a proportion of the number of edges joining nodes inside of $\mathcal{S}$, $l_{\mathcal{S}}$, we can rearrange the inequality to give us a condition based just on the number of edges in these components and their proportions. Let $l_{\mathcal{S}}^{out} = al_{\mathcal{S}}$ with the proportionality constant $a > 0$. Then we can write $d_{\mathcal{S}} = 2l_{\mathcal{S}} + l_{\mathcal{S}}^{out} = (a+2)l_{\mathcal{S}}$, substitute into equation (5) to get

$$\frac{l_{\mathcal{S}}}{m} - \left(\frac{(a+2)l_{\mathcal{S}}}{2m}\right)^2 > 0 \tag{6}$$

and then rearrange to get

$$l_{\mathcal{S}} < \frac{4m}{(a+2)^2}. \tag{7}$$

Now we can decide if a subgraph forms a module based on whether this inequality is satisfied. The problem with that is this inequality contains m, the total number of edges in the network. This therefore means that whether $\mathcal{S}$ is a community or not depends on the size of the whole network rather than just on a comparison between the number of edges inside of the subgraph and the number of edges joining the subgraph to the rest of the network, as we would intuitively expect. This can lead to problems in detecting communities accurately in practice as we will see now.

Let $a < 2$ so that $l_{\mathcal{S}}^{out} < 2l_{\mathcal{S}}$, which is saying that the internal degree of the nodes in $\mathcal{S}$ is greater than their external degree. Substituting this into equation (7) we see that for $\mathcal{S}$ to be a module for all $a < 2$ we need $l_{\mathcal{S}} < m/4$.

Now, we return to our definition of graph $\mathcal{G}$ from the last section again, but in addition this time we want it to have at least three modules with each of the modules defined in $\mathcal{G}$ satisfying both of the inequalities from the last paragraph on their respective subgraph. We will focus on two of these modules, C_1 and C_2 and define the internal links in each of them by l_1 and l_2, respectively. We will define the number of links between them by l_{int} and the number of links between C_1 and C_2 and the rest of the network C_0 by l_1^{out} and l_2^{out}, respectively. Lastly, let $l_{int} = a_1 l_1 = a_2 l_2$, $l_1^{out} = b_1 l_1$ and $l_2^{out} = b_2 l_2$ with $a_1, a_2, b_1, b_2 \geq 0$, $a_1 + b_1 \leq 2$, $a_2 + b_2 \leq 2$ and $l_1, l_2 < m/4$ such that the parameters satisfy the inequalities defined in the previous paragraph.

We now want to compare what would happen if we partitioned the graph in two different ways. More specifically, if we partitioned the graph with C_1 and C_2 in two distinct modules (partition A) or if we partitioned C_1 and C_2 into one module (partition B), both times keeping the subdivision of C_0 the same. We will compare the modularity of each of these partitions to see when it would be beneficial to merge them. Note that since modularity is calculated as a sum over the modules, the modularity generated by C_0 will be the same for each partition and we will denote it by Q_0.

Denoting the modularity of partition A and B by Q_A and Q_B, respectively, we can use equation (4) to get

$$Q_A = Q_0 + \frac{l_1}{m} - \left[\frac{(a_1 + b_1 + 2)l_1}{2m}\right]^2 + \frac{l_2}{m} \tag{8}$$

$$Q_B = Q_0 + \frac{l_1 + l_2 + a_1 l_1}{m} - \left[\frac{(2a_1 + b_1 + 2)l_1 + (b_2 + 2)l_2}{2m}\right]^2. \tag{9}$$

Then defining $\Delta Q \coloneqq Q_B - Q_A$, we have

$$\Delta Q = \frac{2m a_1 l_1 - (a_1 + b_1 + 2)(a_2 + b_2 + 2) l_1 l_2}{2m^2}. \tag{10}$$

We have defined C_1 and C_2 to be modules individually therefore we would expect to have $\Delta Q < 0$ so they are separated. Using equation (10) we find that $\Delta Q < 0$ if

$$l_2 > \frac{2m a_1}{(a_1 + b_1 + 2)(a_2 + b_2 + 2)}. \tag{11}$$

Clearly, it is possible to choose values of the parameters such that this inequality is not satisfied. We know that $0 \leq a_1, b_1, a_2, b_2 \leq 2$ and as long as $1 \leq l_1, l_2 < m/4$ we can take l_1 to be as small as we want it to be, therefore not satisfying the inequality. This means that $\Delta Q > 0$ and maximising modularity would thus be unable to resolve these smaller community structures, which we planted.

For a more concrete example, consider the case where C_1 and C_2 have the same number of internal links $l_1 = l_2 = l$. Furthermore, let only one edge join each module to the network and one edge join the modules together. That implies that $a_1 = a_2 = b_1 = b_2 = 1/l$. Using equation (10) we now see that we have $\Delta Q > 0$ and therefore C_1 and C_2 cannot be resolved by modularity maximisation when $l < l_R^{min} = \sqrt{m/2} - 1$.

To reiterate, these could be fully connected modules separated by a single link. The results are similar when $l_1 \neq l_2$ but just scaled differently [6].

Although we will not derive it here, Fortunato and Barthélemy [6] then go on to derive an inequality on a network which is a ring of k cliques of v nodes. To clarify, a clique of v nodes is v nodes with links joining every node in the clique to every other node in the clique. Each of these cliques then has 2 external links to other cliques such that the cliques are all connected and can form a ring-like structure (see figure 5). By comparing the modularities of when each clique is a cluster to when each adjacent pair of cliques is a cluster we find that $Q_{single} > Q_{pairs}$ only if $\sqrt{m} = v(v-1) + 2 > k$. Since v and k are independent we can easily find values of them such that the inequality is not satisfied and therefore modularity maximisation would suggest that adjacent pairs of cliques are members of the same cluster. This is however not surprising since if the size of a network increases while keeping the average degree of a node similar then the expected number of links between nodes will decrease. This means that even a small number of edges joining subgraphs will be unexpected and cause an increase in modularity when viewed as two clusters rather than one [5].

This is a problem with the model in that the choice of the null model governs the expected number of links in a subgraph. The null model assumes that any node can connect to any other node in the network and that each stub would connect to any other stub with equal probability. In real-world networks we know that this is not true and even random graphs drawn from the null model, which we know do not exhibit community structure, will be found to have some. This is due to the random sampling fluctuations [7].

A second problem is that if we are to distinguish between an optimal partition and an intuitive partition is we would have to assume an external definition of what an intuitive partition is. In fact, the partition we find as intuitive may not be the same partition we think is intuitive next time we look at the graph. It is therefore hard to define something mathematically that is quite a vague notion anyway [7].

We can now see that the definition of community suggested by modularity is not consistent with its optimisation as smaller communities may be merged in favour of larger ones. It is therefore impossible to tell whether certain detected modules are in fact a combinations of smaller modules, which casts doubt over the effectiveness of modularity maximisation for community detection [6]. Nonetheless, methods have been developed to deal with this by allowing us to look at the community structure in graphs at many different resolution levels meaning we can modify our modularity function to favour smaller or larger communities. In the next subsection, we will look at one of these.

3.2 Multiresolution Methods

Here, we present a method for overcoming the resolution limit by following the paper Analysis of the Structure of Complex Networks at Different Resolution Levels [1]. As we know from the previous subsection, two identical clusters connected only by a link and connected to the rest of the network only by a link each will not be individually resolved by modularity maximisation if the number of internal links in each module $l < l_R^{min} = \sqrt{m/2} - 1$. Hence if we find a way to decrease the right-hand side of this inequality we can resolve smaller modules. We can do this using weighted networks.

The modularity function for a weighted network is given by

$$Q^* := \sum_{c=1}^{n_c} \left[\frac{w_{cc}}{w} - \left(\frac{w_c}{2w} \right)^2 \right] \tag{12}$$

where we are summing over n_c modules with w_{cc} being the internal strength of module C_c and w_c being the total strength of module C_c. That is, the sum of the weights of edges connecting nodes in C_c and the sum of the strength of each node in C_c, respectively, where the strength of a node is the sum of the weights of the edges connected to it. Lastly, $2w$ is total strength, the sum of the strengths of all nodes in the network.

In a similar way to last subsection, we can show that weighted networks give an analogous inequality for when the two modules defined above will not be resolved in $w_{cc} < \sqrt{w/2} - 1$. We can however decrease the right-hand side of this inequality if we increase the strength of each node by a quantity λ, giving us

$$w_{cc} < \frac{1}{2}(\sqrt{2w + n\lambda} - N_c\lambda - 2) \tag{13}$$

with N_s being the number of nodes in cluster C_c and n being the total number of nodes in the network. We want to keep the structural characteristics of the new network as similar as possible to the original network. It turns out that a good way to do this is by just adding a self-loop of weight λ to every node while setting all the previous edges' weights to 1. This does not affect the strength distribution of the original edges which define the network topology. Furthermore, it affects each node the same way. We call the shifted network $\mathcal{G}_\lambda$ when the network is shifted by λ.

Now, since we can modify the inequality (13) by varying λ and both the terms containing λ scale differently, we can make it such that we can resolve any size module we want in the current graph we are working with by using modularity maximisation. In fact, the modularity function of the graph $\mathcal{G}_\lambda$ becomes

$$Q_\lambda = \sum_{c=1}^{n_c} \left[\frac{2w_{cc} + N_c\lambda}{2w + n\lambda} - \left(\frac{w_c + N_c\lambda}{2w + n\lambda} \right)^2 \right]. \tag{14}$$

Thus the modularity function that we were working with previously corresponds to $\lambda = 0$. However, if $\lambda > 0$ maximising the modularity reveals smaller groups of nodes, and if $\lambda < 0$ we will find larger groups of nodes. This is difficult to see mathematically but is a result of the way the terms in the summand of (14) scale with λ. Hence we will instead show this heuristically in figure 1.

The LFR benchmark graphs are graphs created by an algorithm which allows us to choose parameters such as the number of nodes and minimum community size so that we can generate graphs with *a priori* known community structure. As such we can guarantee we will have communities of various sizes.

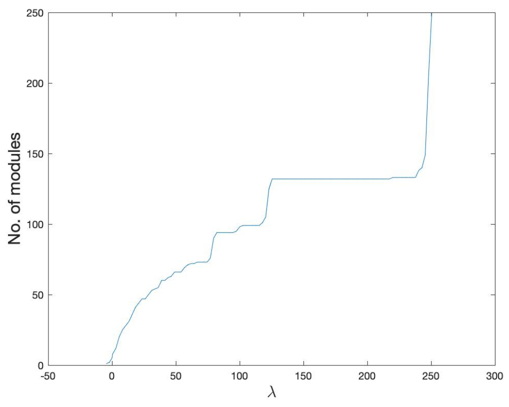


Figure 1: Number of modules obtained by optimising Q_λ using Tabu search on the LFR network described in figure 2a. This was generated using NetworkX to create the network, Radatools 5.2 software [4] to perform the optimisation and Matlab to plot it.

We see that varying λ allows us to split the networks into many different partitions from the partition with 1 module all the way up to when each node is in its own community. Nonetheless, in general, these structures will not be hierarchical as nothing implies that the substructures found by varying the resolution parameter would be hierarchical. Naturally, the next question to ask would be which value of λ provides the most information about the structure of the network and does a value of λ exist that allows us to see the whole community structure? This is what we will investigate in the next section.

3.3 Problems with Multiresolution Methods

The paper Limits of Modularity in Community Detection [9] proves that there are even problems with multiresolution methods. When the resolution parameter is low there is a tendency to merge small subgraphs whereas when the resolution parameter is high there is a tendency to split large subgraphs. The paper proves that for a different multiresolution technique to the one we have and for certain graphs we will not be able to find λ such that neither of the aforementioned issues occur. More specifically, we will not be able to avoid either the splitting of random subgraphs or the merging of cliques which are loosely connected.

The authors do not perform analysis on our specific multiresolution model but they ran numerical simulations to back up this perspective. In fact, they show that on some graphs with $10,000$ nodes and planted clusters varying in size from 10 to 1000 nodes even the resolution parameter which gives the best results does not classify all the known community structures very accurately.

Thus the simultaneous elimination of both issues is not possible. This is not to say that multiresolution methods are not useful and we can gain a lot of information about the community structure by looking at it at different resolution levels. In figure 1 we see that certain partitions are stable in the sense that they persist for a significant number of values of λ. Furthermore, in synthetic graphs these stable partitions tend to be over the intuitive partitions we would expect to get [1]. That said, we cannot invert this argument directly and say that a stable partition is definitely the partition we want or that an unstable partition is unimportant. In fact, if we did say a certain scale is more important we would be suggesting that it is better to optimise a different objective function than our standard modularity. Instead this method is intended to allow us to investigate the community structure at different resolution levels and use our intuition and what we know about the problem to make an informed decision on the community structure [1].

Now we have introduced the resolution limit and looked at a possible way to deal with it we will have a look at another problem with modularity, degeneracies in the modularity function. These are again implicit in the definition of modularity, but methods have been developed to help deal with them.

4 Degeneracy Among High-Modularity Partitions

4.1 Introduction to Degeneracy

When we say that the modularity function exhibits degeneracies we mean that there are a large number of high-modularity partitions often without a clear global maximum. This occurs because even when merging clusters is not beneficial the size of the change in modularity for doing it can be small, so the modularity is not greatly affected. Then, as we have more modular structures, the ways to combine them grows exponentially hence these problems combine to give us degeneracies in the modularity [7]. On one hand, we should easily be able to get a high-modularity partition for many techniques. On the the other hand the maximum is practically unreachable and some of these partitions may differ significantly from the partition with highest modularity [9].

As a more concrete example let $\mathcal{G}$ be the network defined in section 2.1, this time with k sparsely interconnected groups of nodes in which each group has roughly the same total degree $d_c \approx 2m/k$. Suppose m is small enough such that the partition with the highest modularity is the one grouping each cluster individually, by using equation (4) we see that the change in modularity for merging a pair of these clusters C_i and C_j is given by

$$\begin{aligned}\Delta Q &= \frac{l_i + l_j + l_{int}}{m} - \left(\frac{d_i + d_j}{2m}\right)^2 - \left(\frac{l_i}{m} - \left(\frac{d_i}{2m}\right)^2 + \frac{l_j}{m} - \left(\frac{d_j}{2m}\right)^2\right) &(15)\\ &= \frac{l_{int}}{m} - 2\left(\frac{d_i}{2m}\right)\left(\frac{d_j}{2m}\right) &(16)\end{aligned}$$

where the variables are defined as they were in the section 2.1. To reiterate l_{int} is the number of links between module i and module j and l_i and l_j are the number of internal links in module i and j, respectively.

Equation (16) then shows that ΔQ is bounded below by $-2k^{-2}$. Hence, with only 20 groups of nodes the change in modularity is bounded below by -0.005 and as k increases the size of this penalty will only get smaller and it will become increasingly difficult for modularity maximisation techniques to determine the optimal partition from these suboptimal alternatives [7].

Furthermore, the number of competitive alternative partitions grows combinatorially with the number of modular structures k. We can find the lower bound of it by considering the connected modular network with the fewest inter-module edges and finding the number of ways to split the groups into connected components. That is, we want the number of ways to cut the inter-module edges of a string of k connected components, where a string is a ring with one inter-module edge removed. Thus there are $k - 1$ inter-module edges each which can either be cut or not so there are 2^{k-1} possible choices and the number of competitive modular structures is bounded below by 2^{k-1} [7].

Here, we have considered only modular networks but it is important to note that hierarchical networks exhibit at least as many degenerate solutions as simple modular networks and the alternative modularity score can be even closer to the optimum. Intuitively, this is because each level of the hierarchy has its own modular structures which can be merged between the same or different levels of the hierarchy structure. This therefore gives us even more alternative solutions so we can continue to use the previous lower bound of 2^{k-1} competitive alternatives, but this time where k represents the number of modular structures at the lowest level of the hierarchy [7]. We note that these problems even continue to occur for our multiresolution method presented last section [7].

As mentioned at the start of the paragraph, one of the problems with having lots of competitive partitions is that finding the partition of optimal modularity becomes practically impossible. As such, when we try to optimise modularity using stochastic algorithms, such as the greedy algorithms mentioned in section 2.1, they often return different partitions of the same network each time you run them. If these partitions are similar we can create a consensus clustering, a single partition that represents the whole set in a similar way to a mean being a good representation of a collection of numbers. However, if the variation in the partitions is too great just providing one representative cluster ignores much of the information in the same way only providing the mean for a wide range of numbers does. The next two papers we will see will discuss a method to deal with this by providing several representative partitions, each for a different cluster of similar partitions. This will be useful for hierarchical networks as they tend to give many varied partitions when run through stochastic algorithms due to them having many different layers to their structure. Note that the next papers we will discuss are not exclusive to optimising modularity and can in fact be used for any problem which provides a collection of partitions which need to be summarised [8].

4.2 Exploring the Solution Landscape

Since we will now be working with grouping nodes into partitions and grouping partitions into representative groups of partitions we will have to be more specific about the language we use. We will use the word partition for the assignment of nodes to communities and we will use the word cluster to describe the assignment of partitions to their representative group of partitions [8].

Here we present the method given in the paper Exploring the Solution Landscape Enables More Reliable Network Community Detection [2]. As mentioned previously, if we used a stochastic algorithm to maximise modularity we would likely return different partitions of the network each time, which can often be quite varied. We could study the landscape of the modularity for all partitions if we wanted no wasted information but it would take too long to find the modularity of and plot all partitions. Once we had the landscape we would then have to explore it visually to decide which cluster is most representative. Here, we present a method for clustering partitions that should not only cluster the partitions but also decide when we have enough partitions to describe the solution landscape accurately so we can stop searching for more. We can then select the cluster we wish based on which contains the highest modularity partition or the most partitions in its cluster.

The first thing we will need to describe the solution landscape is the notion of distance. We will not go in to that too deeply here as it is not too relevant to what we are doing and, in fact, different functions can be used. The one used in the paper, and that we will use in our code, is useful for us as it works for hard and hierarchical communities.

Of much more interest to us is the clustering algorithm as that is where the interesting modelling ideas are. Many clustering algorithms generally involve NP-hard optimisation problems, however this algorithm gives a fast and transparent approach to clustering. More so, the algorithm decides when is safe to stop by itself and does not require you to specify the number of clusters thus mitigating the ambiguity caused by multiple solutions. Lastly, having an interpretable coarse-grained landscape simplifies further analysis as it provides us with a much clearer description of what is happening.

Given a set of partitions and a partition distance threshold d_{max} the clustering algorithm then works like so:

1. Order all p network partitions from highest to lowest quality. For us this will be judged by the modularity.

2. Let the highest quality network partition form cluster center 1.

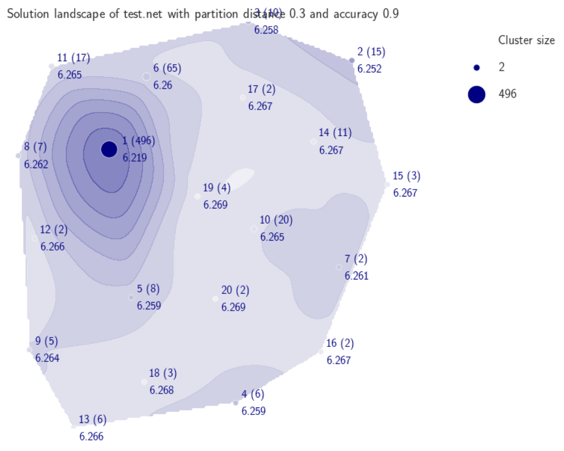
3. Repeat this step until all network partitions have been clustered. Among the not-yet-clustered partitions, pick the one with the highest quality and assign it to the first of the q cluster centers that it is closer to than d_{max}. If no such cluster center exists, let it form cluster center $q + 1$ [2].

We will refer to the qth generated cluster as the qth best cluster and the ith highest modularity partition in it as the ith best partition in that cluster. Note that the distance threshold governs the resolution of the coarse-grained landscape in that decreasing the threshold will produce more clusters containing more similar networks and increasing it will do the opposite. We will see this in action but first we will describe how we know when we can terminate our algorithm.

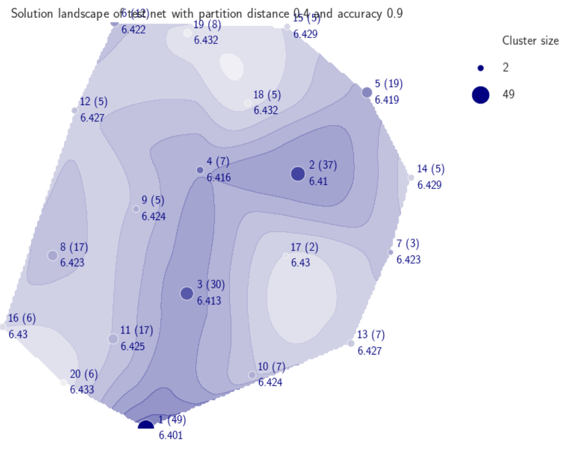
To terminate our algorithm we want the solution landscape to be complete: its coarse-grained structure should be marginally affected by new partitions. It should therefore require fewer partitions to find a complete landscape of a network with a clear community structure than one with a diffuse community structure as the latter will have fewer similar partitions. Ultimately, we want the probability of a new partition to be clustered into an existing cluster to be close to 1. We will use this as a validation score to check when the solution landscape is complete. In our code, we will define the stopping criteria by splitting all the partitions we have into a training set and a validation set, applying the partition clustering algorithm to the training set then measuring the validation score on the validation set. If the validation score is greater than a specified accuracy level (usually around 0.9) we can terminate the algorithm, if not then we use our stochastic algorithm to find more partitions and repeat the process. To avoid problems cause by the stochasticity of these methods we use at least 100 partitions in our validation set or $p/2$ partitions when we have fewer than 200 partitions overall so we get a more reliable validation score.

The authors of this paper gave a link to the code they developed to carry out this algorithm in their paper [2, 12] so we will use this. The Python package Networkx can then be used to generate graphs to apply their method to. Here, we will vary the network used and the distance threshold to see how the community landscape is affected empirically. We will then also plot alluvial diagrams to see how the partitions vary within and between clusters. We could then draw these networks but we will save that for the next subsection as here the solution landscape is most informative of the model.

We will test our network on LFR benchmark graphs while varying the mixing parameter μ. This is the average fraction of neighboring nodes of a node that do not belong to any community that the benchmark node belongs to so we would expect



(a) Parameters: $n = 250$, tau1 $= 3$, tau2 $= 1.5$, mu $= 0.1$, average_degree $= 5$, min_community $= 20$, seed $= 10$ and $d_{max} = 0.3$



(b) Parameters: $n = 250$, tau1 $= 3$, tau2 $= 1.5$, mu $= 0.2$, average_degree $= 5$, min_community $= 20$, seed $= 10$ and $d_{max} = 0.4$

Figure 2: The solution landscapes of two LFR graphs created using the LFR_benchmark_graph function on NetworkX with parameters given above. Each point represents a cluster of partitions. The first number is q if it was the qth cluster to be generated and the number in brackets is the number of partitions in the cluster. The number below is the “length” of the best partition which is the function Infomap minimises to find the optimal partition. The length of the best partition is what the colour coding is based upon.

to see a more complex landscape as this increases since the community structure will be more diffuse.

In figure 2 we used an algorithm called Infomap [3] to generate partitions of the network then plotted the first 20 clusters that are generated by the algorithm once it has finished running so we can visualise some of the structure of the solution landscape. The first thing to note about these plots is that the network used in figure 2b has a higher mixing parameter and its landscape has a more complex structure (including more than one local minimum) as we expected. The second thing to note is that we have had to use a greater distance threshold when we are using a greater mixing parameter because the algorithm was taking too long to run when $d_{max} = 0.3$. This shows that networks with more diffuse community structures take more time for the algorithm to work on as there are more different local minima that the stochastic partitioning algorithm can produce. Furthermore, we see that increasing the distance threshold makes the clustering algorithm run faster since less similar partitions end up in the same clusters so the validation score gets greater faster.

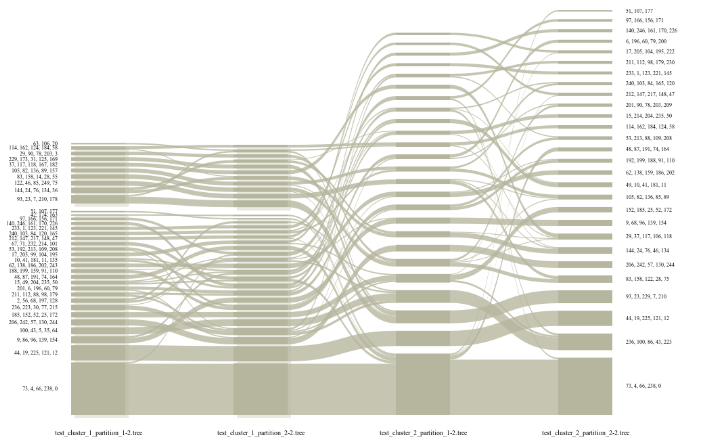
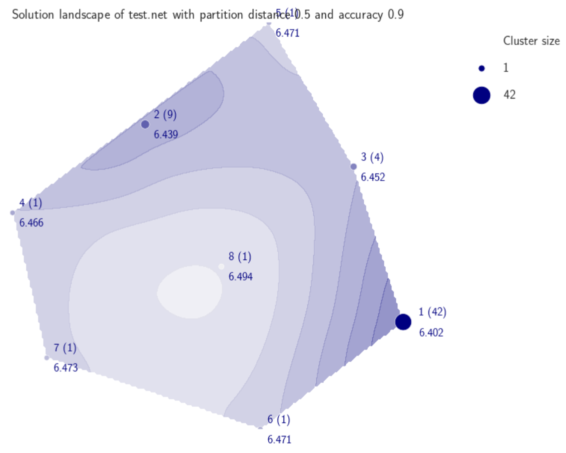


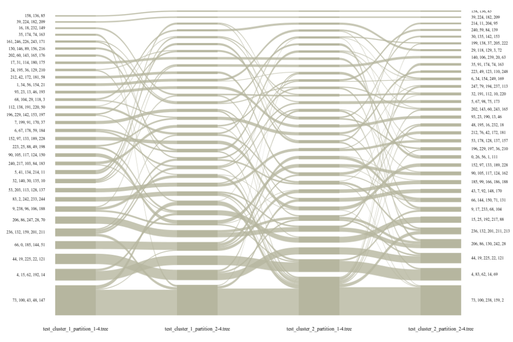
Figure 3: Alluvial diagram for the LFR network used in figure 2a. Generated using [3].

Alluvial diagrams can provide information on qualitative pairwise differences between partitions. Each partition is represented as a stack with each rectangle being a module. Streamlines then connect these rectangles showing how nodes change modules between partitions with the streamline thickness being proportional to the flow of nodes. In figure 3 we see how the centre partition of the best cluster splits as we go to the next best partition in the cluster (according to Infomap), then how this partition changes as we go to the best partition of the second best cluster and lastly as the partition changes as we go to the second best partition of the second best cluster. The important features of this are that although the first two partitions in the diagram look like they are changing quite a bit in actual fact the nodes are just changing modules within each of two larger modules therefore showing we have detected a hierarchical structure. The second thing to note is the larger change in how the nodes are partitioned as we go from the best cluster to the second best. This is because partitions which are similar to the best cluster would not be in it hence we expect a larger change.

Figure 4 allows us to see the effect increasing the distance threshold has on the clustering and partitioning. Here, we use the LFR benchmark graph from before with $\mu = 0.2$ but this time we set $d_{max} = 0.5$ rather than 0.4. Comparing these plots to the ones in figure 2b and figure 3 we see the landscape becomes much less complex as partitions which are not as similar are grouped together. This is then shown in



(a) Solution landscape



(b) Alluvial diagram. Generated using [3].

Figure 4: Plots generated using the LFR benchmark graph as described above with $\mu = 0.2$ and $d_{max} = 0.5$.

our alluvial diagram by there being much larger changes in the partition between the best and second best partitions in each cluster compared to figure 3.

In the next section we will see a similar method for finding representative partitions of a set of partitions but this time the method will take an information-theoretic perspective so as to avoid the choice of distance between partitions and the choice of distance threshold.

4.3 An Information-Theoretic Perspective

The paper Representative Community Division of Networks [8] provides an alternative perspective on how to deal with sets of multiple partitions arguably making improvements on the method presented last subsection. We do not have enough space and therefore will not go too deeply into the detailed mathematics of how this works however we will look at the intuition behind it as that is where the most of the interesting modelling ideas lie. We will then provide plots generated numerically using the examples from the paper as the author sent me the code he used to generate the plots. We will vary parameters comparing the results we get and we will investigate a problem we discussed earlier in the paper to see how this method deals with it.

Again, this paper works on the premise of clustering similar partitions into groups and identifying a partition to represent each group. The difference is that this method uses an information-theoretic approach: it employs the minimum description length principle. This proposes that when selecting between models of a dataset, the best model is the one which represents the data most succinctly. Thus, applied to our problem, we want to transmit a set of partitions to a receiver and minimise the amount of information required for the transmission.

The most obvious way to transmit these partitions would be to transmit them one-by-one representing the community labels as numbers or letters. However, we can make use of the fact that many of these partitions will be similar to reduce the amount of information we would need to transmit. We can divide our network into clusters of similar partitions, find a partition that represents each cluster (a mode) and then describe the remaining partitions by how they differ from these modes. This means that the only partitions we have to transmit in full are the modes therefore decreasing the total amount of information we have to transmit.

Suppose we want to divide the set D of p partitions into K clusters. To do this we want to submit K representative partitions which are themselves members of D then for each individual partition we need to say which cluster it is in and how it differs from the mode. The more similar a partition is to its chosen mode the less information will have to be transmitted thus choosing a mode that represents all partitions in the cluster well reduces the overall information submitted. We therefore use this as the criterion to derive the best set of modes in turn applying the maximum description length principle to our problem.

Ultimately, we can represent the total description length $\mathcal{L}_{total}$ as the sum of three terms: the amount of information required to transmit the modes, the amount of information needed to specify the cluster of each partition and the amount of energy needed to define the individual partitions in terms of their modes. Each of these terms are defined in the paper but we will not do that here. Instead, we ask ourselves how the number of clusters K is chosen.

In general, we want a smaller number of clusters, no more than about 10 for the plot to be more intuitive. A natural way to do this is by including a penalty term in the objective function such that we penalise too great a number of clusters. This can be done by defining $\mathcal{L} := \mathcal{L}_{total} + \lambda K$ for some $\lambda \in \mathbb{R}$ as our new objective function to minimise. As a result each extra cluster provides an extra penalty of λ. Choosing λ too large or too small will give inefficient descriptions of the data because some partitions will not be similar to any of the modes or because too much information will be transmitted describing the modes, respectively. The paper finds that $\lambda = 1$ works well in many cases. In the paper and our code, the objective function is then minimised by means of a greedy algorithm, although there are other methods available.

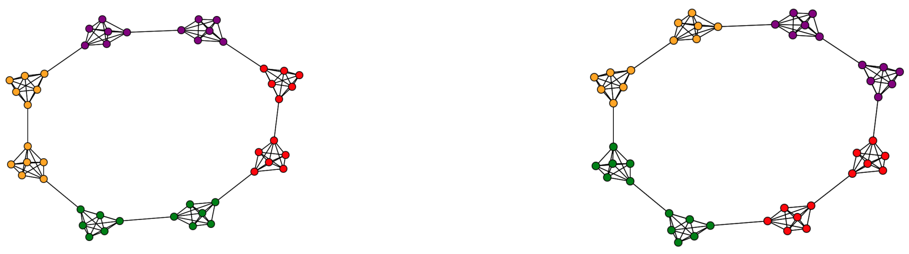


Figure 5: The modes generated when using $\lambda = 1$ on 1000 partitions of this ring network with the partitions generated as described in this subsection.

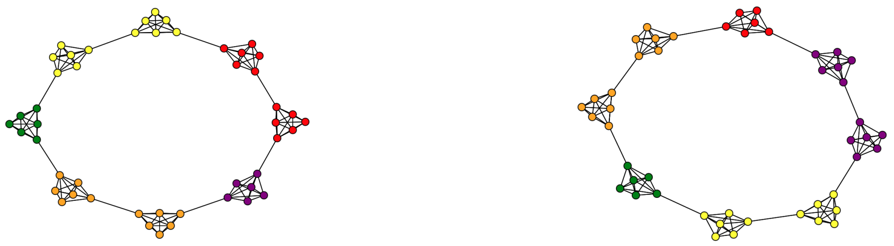


Figure 6: The modes generated when using $\lambda = 0.5$ on 1000 partitions of this ring network with the partitions generated as described in this subsection.

Now we have the method, let us look at some examples and how varying the parameter λ affects the partitions generated. As we said earlier in the subsection, we will adapt the code sent to us by the author of the paper to get the plots. This code gets its partitions by sampling them from the posterior distribution of a fitted statistical model.

We will apply the new method to a ring of cliques, like we discussed in section 3.1. The community detection algorithm we use to generate the partitions in the code also suffers at times from grouping pairs of cliques as a single module. Hence, in figure 5 we see that when $\lambda = 1$ we have two modes, both of which corresponding to the rotationally invariant configurations generated if we group adjacent cliques of nodes into communities.

Although, the community detection algorithm does also sometimes group cliques individually. Letting $\lambda = 0.5$ the algorithm now generates 9 modes, many with 3 pairs of cliques as modules and then 2 cliques individually in modules as shown in figure 6. This was to be expected as the penalty for having more clusters is not as harsh as before.

In this section we have introduced the problem of degeneracies in the modularity function and seen two alternate methods for dealing with it but both built around completely different perspectives. This is interesting from a modelling perspective and great to compare as each method has its own advantages and disadvantages. The method built upon examining the solution landscape is useful in that it provides a stopping criteria so you know when you have enough partitions to make a reliable estimate of the landscape and the representative partitions. Furthermore, the solution landscape also gives a good visual method for determining how the clusters and partitions compare and it can in turn simplify further analysis as we can just compare cluster centres rather than partitions. The method built upon information-theoretic principles then provides a completely different perspective but, in doing so, relinquishes the need for a partition distance function and it does not depend on the number of input partitions provided the space is well sampled.

5 Conclusion

We set out this essay with the intention of investigating how modularity maximisation could be used in community detection, its disadvantages and how models have been developed to mitigate these disadvantages. The first thing we did was introduce community detection and modularity and describe how modularity maximisation can be used to detect communities within networks. The first problem we discussed in detail was the resolution limit. We explained the problem and went through some examples to show when it occurs analytically. We then discussed why it occurs considering problems with the model of modularity and our definition of what an intuitive partition is. In the next subsection, we introduced a multiresoltuion method for dealing with the resolution limit problem before going on to explain why there even exists problems with multiresoltuion methods in the next subsection. The next section was centred around the modularity function exhibiting extreme degeneracies. First we introduced the problem then we showed when it occurs with an easy example before finally getting in to two recently developed methods to deal with it. These methods both considered the problem from completely different perspectives and provided two interesting results which each have their own benefits.

Throughout the essay we used numerical simulations to confirm our findings often expressing the results using plots. We even used the multiresoltuion and the landscape method on the same LFR benchmark graph to get an interesting comparison. More so, we explored the ring of cliques analytically in the early sections before returning to it with the information-theoretic method developed in the final paper to see how methods have developed to deal with these fundamental problems of modularity.

References

[1] Alex Arenas, Alberto Fernandez, and Sergio Gomez. Analysis of the structure of complex networks at different resolution levels. *New J. Phys. 10 (2008) 053039*, 2:1–15, Jan 2008. URL `https://arxiv.org/abs/physics/0703218`.

[2] Joaquín Calatayud, Rubén Bernardo-Madrid, Magnus Neuman, Alexis Rojas, and Martin Rosvall. Exploring the solution landscape enables more reliable network community detection. *Phys. Rev. E*, 100:1–6, Nov 2019. URL `https://link.aps.org/doi/10.1103/PhysRevE.100.052308`.

[3] D. Edler, M. Rosvall, and A. Eriksson. The mapequation software package. `mapequation.org`. Accessed on 5th April 2022.

[4] Alberto Fernandez and Sergio Gomez. The radatools 5.2 software package. `https://deim.urv.cat/~sergio.gomez/radatools.php#authors`. Accessed on 7th April 2022.

[5] Santo Fortunato. Community detection in graphs. *Physics Reports 486, 75-174 (2010)*, 2:1–16, 27–41, Jan 2010. URL `https://arxiv.org/abs/0906.0612v2`.

[6] Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. *Proc. Natl. Acad. Sci. USA 104 (1), 36-41 (2007)*, 2:1–6, July 2006. URL `https://arxiv.org/abs/physics/0607100`.

[7] Benjamin H. Good, Yves-Alexandre de Montjoye, and Aaron Clauset. The performance of modularity maximization in practical contexts. *Phys. Rev. E 81, 046106 (2010)*, 2:1–12, Apr 2010. URL `https://arxiv.org/abs/0910.0165`.

[8] Alec Kirkley and M. E. J. Newman. Representative community divisions of networks. *CoRR*, 2:1–7, Feb 2022. URL `https://arxiv.org/abs/2105.04612`.

[9] Andrea Lancichinetti and Santo Fortunato. Limits of modularity maximization in community detection. *Physical Review E84, 066122 (2011)*, 2:1–8, Feb 2012. URL `https://arxiv.org/abs/1107.1155`.

[10] M. E. J. Newman. Fast algorithm for detecting community structure in networks. *Phys. Rev. E 69, 066133 (2004)*, 1:1–4, Sep 2003. URL `https://arxiv.org/abs/cond-mat/0309508`.

[11] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E 69, 026113 (2004)*, 1:7–8, Aug 2003. URL `https://arxiv.org/abs/cond-mat/0308217`.

[12] Martin Rosvall and Anton Eriksson. Solution landscape. `https://github.com/mapequation/solution-landscape`, April 2020. Accessed on 5th April 2022.